Universidad Carlos III de Madrid
Escuela Politécnica Superior

ITT. Sistemas de Telecomunicación

Departamento de Tecnología Electrónica

Proyecto Fin de Carrera

"Creación y Puesta a Punto de Toolchain GNU Para Microprocesadores ARM7"

Autor: Juan Manuel Rivera Panadero

Tutor: Michael Victorio García Lorenz

Septiembre 2010

Título: “Creación y puesta a punto de toolchain GNU para microprocesadores ARM7”

Autor: Juan Manuel Rivera Panadero

Tutor: Michael Victorio García Lorenz

EL TRIBUNAL

Presidente: _____

Vocal: _____

Secretario: _____

Realizado el acto de defensa y lectura del Proyecto Fin de Carrera el día __ de _____ de 20__ en Leganés, en la Escuela Politécnica Superior de la Universidad Carlos III de Madrid, acuerda otorgarle la CALIFICACIÓN de

VOCAL:

SECRETARIO: PRESIDENTE:

Agradecimientos

En estas líneas quiero agradecer a toda mi familia por haberme apoyado tanto a lo largo de la carrera y por la paciencia que han tenido.

A Natalia, mi novia, por los ánimos que me ha dado durante toda la carrera y la paciencia que ha tenido por el tiempo que no he podido estar junto a ella.

También a mis compañeros de clase que parte de este trabajo es gracias a ellos, en especial a Juan Carlos, ánimo que ya queda menos.

Y en general a todas las personas que me han apoyado y que están a mi lado.

Índice:

1. Introducción

El principal objetivo de este proyecto consiste en crear una toolchain que permita sustituir las herramientas comerciales de Keil utilizadas hasta ahora para la programación y depurado de programas en procesadores de la familia ARM7 por herramientas de código abierto GNU. Para ello, se pretende detallar tanto la instalación de las herramientas GNU, la configuración del entorno de desarrollo (Eclipse) y la creación de los archivos necesarios para la correcta adaptación del procesador a la nueva toolchain.

Uno de los objetivos secundarios es generar una plantilla que permita inicializar el procesador de forma sencilla, explicar el significado de Inicializar.s, qué ocurre cuando se pulsa sobre “Build Project”, cómo se escribe el programa en la memoria del procesador, etc.

Otro de los objetivos secundarios consiste en poner en práctica los conocimientos adquiridos a lo largo de la carrera, profundizando sobre el manejo de procesadores a un nivel más detallado que no consiste únicamente en grabar el programa en la memoria flash, sino interesarse por la arquitectura del procesador y cuáles son sus características de funcionamiento.

1.2 Motivación

El motivo principal por el cual se utilizan herramientas de software libre es porque son de bajo coste, punto que hay que tener en cuenta debido a los altos costes de los programa de desarrollo profesionales. Pero debido a que son más baratas que las herramientas profesionales, presentan algunos inconvenientes que las herramientas profesionales no presentan, por ejemplo son más complicadas de instalar, de utilizar y la interfaz con el usuario es menos intuitiva. En el lado positivo, se puede decir que el compilador GNU es mejor que el ofrecido por las herramientas profesionales, ya que al ser software libre su capacidad de evolución y mejora es mucho mayor y su código estará mucho más depurado.

Otros de los motivos por los que es recomendable el uso de esta toolchain GNU, es que si hasta ahora se ha estado utilizando la versión de prueba que ofrece Keil para manejar el procesador, los proyectos creados no pueden sobrepasar el límite de 32 Kb impuesto para la versión de evaluación de uVision. Esta restricción desaparece y se podrán cargar programas de mayor tamaño gracias a la utilización de las herramientas GNU.

1.3 Estructura del documento

A continuación se enumerarán los distintos capítulos contenidos en el documento junto a una breve descripción de su contenido:

En el capítulo 1, introducción, se realiza una breve descripción de lo que se va a conseguir una vez finalizado el proyecto y cuáles son los objetivos a cumplir. También se incluye un apartado en el que se indica la estructura del documento.

En el capítulo 2, material necesario, se va a detallar cuál es el material hardware y software necesario para el correcto funcionamiento de la toolchain GNU además una descripción de cual es la funcionalidad de cada herramienta.

En el capítulo 3, descarga de las herramientas de trabajo, se indica de donde y como hay que descargar las herramientas software necesarias.

En el capítulo 4, instalación de las herramientas de trabajo, se explica detalladamente como hay que instalar cada herramienta software.

En el capítulo 5, configuración del entorno de trabajo, se explica cuales son los pasos necesarios para configurar Eclipse para poder ejecutar las herramientas externas necesarias.

En el capítulo 6, creación de proyecto con Eclipse, se indica que hay que hacer para crear un nuevo proyecto y como añadir archivos al proyecto. También se realiza

una descripción sobre los archivos que son necesarios para el correcto funcionamiento de la toolchain.

En el capítulo 7, ejemplo práctico, se va a realizar un ejemplo en el cual se demostrará el correcto funcionamiento de la toolchain. Se explicará el contenido del fichero “main.c” y “función_encendido.c”, también se va a explicar como compilar el proyecto para luego grabarlo en la memoria flash y como depurarlo.

En el capítulo 8, conclusiones, se exponen las conclusiones obtenidas por el autor.

En el capítulo 9, trabajo futuro, se va a indicar las posibles vías de desarrollo de este proyecto.

En el capítulo 10, Diagrama de Gantt y presupuesto, se va a detallar cuales han sido las tareas realizadas en el proyecto, su duración y su representación sobre el diagrama de Gantt, además se incluirá el presupuesto del proyecto.

2. Material necesario.

Para la realización de este proyecto, es necesario el uso de material software y hardware.

2.1 Material hardware.

A continuación, se muestra el material hardware necesario para el funcionamiento del entorno de trabajo:

- Ordenador con al menos 2 puertos USB.

- Placa de desarrollo con microprocesador LPC2148: Para la realización de este proyecto, el procesador que se va a emplear es el LPC2148, que pertenece a la familia de procesadores ARM7. Este modelo en especial posee una memoria RAM de 32 Kb y una memoria flash de 512 Kb.

- ARM USB JTAG:

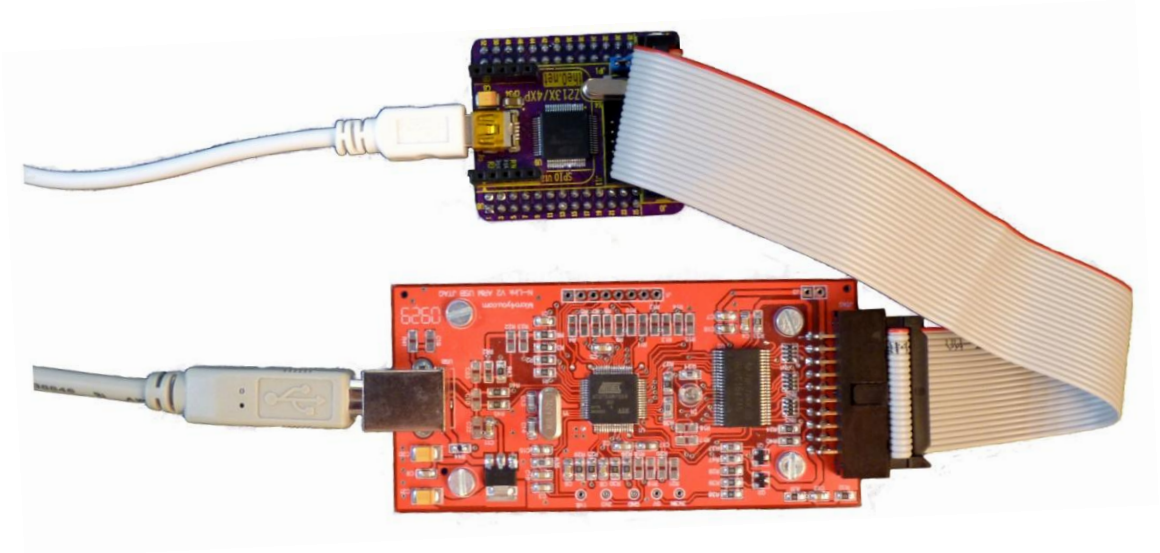
- Cable plano conector idc 20-pin:

- Cable de alimentación del JTAG de tipo miniUSB:

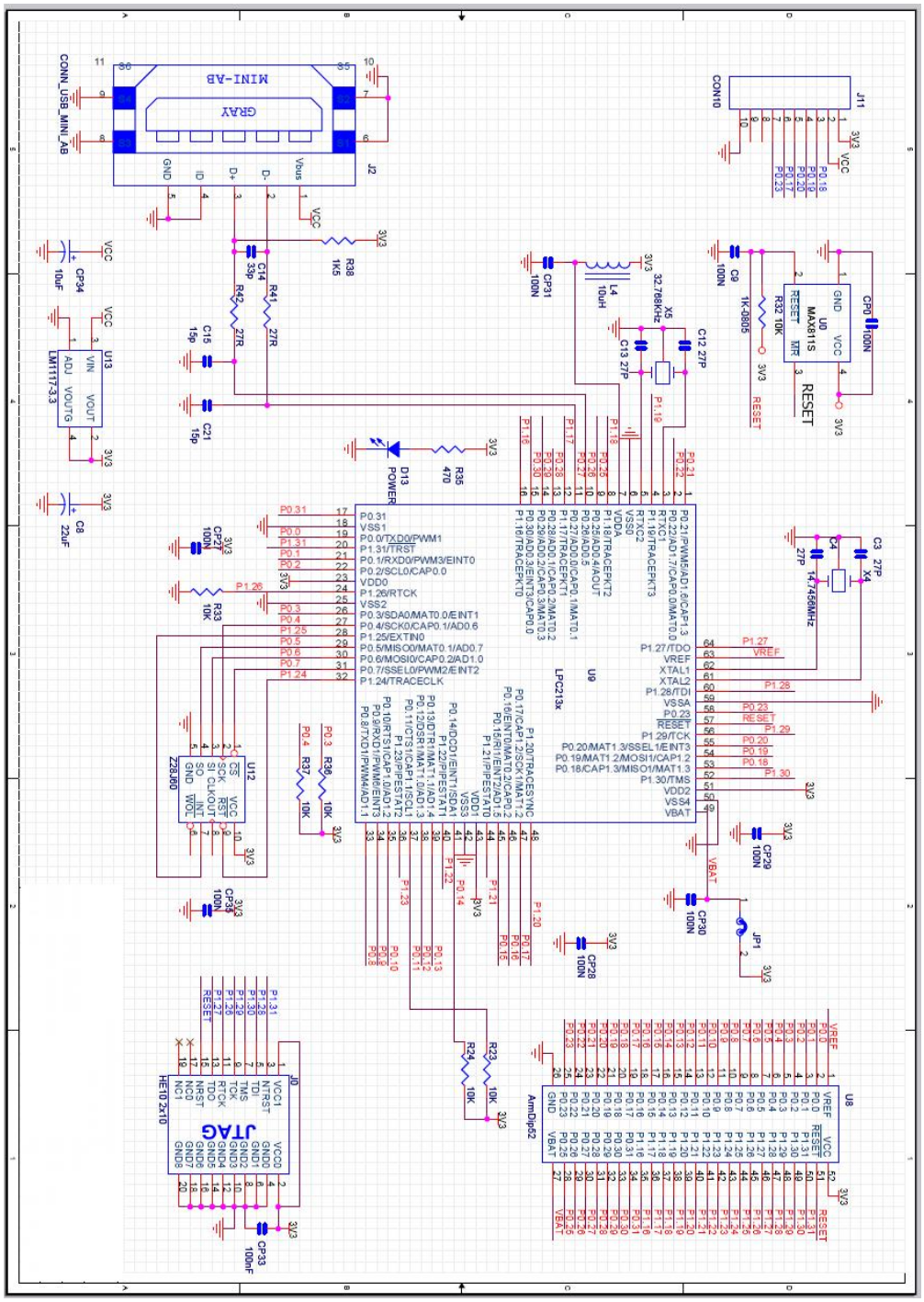
- Cable de alimentación de la placa de desarrollo USB tipo B:

El conexionado de los elementos hardware es muy sencillo. Hay que conectar el cable plano de 20 pines entre la placa de desarrollo y el JTAG. No cabe posibilidad alguna de equivocación en la posición en la que debe ir, puesto que tanto el conector macho como el hembra en los dos extremos del cable tiene una ranura que indica lo posición correcta de conexión.

Posteriormente hay que conectar ambos cables de alimentación a los puertos USB del ordenador. El cable miniUSB ha de conectarse en la placa de desarrollo del micro y el cable USB tipo B se conecta con el JTAG, de forma que todo ha de quedar como se muestra en la siguiente imagen.



Para finalizar, en la imagen inferior se muestra el esquemático de la placa de desarrollo del procesador. Este muestra los distintos pines multipropósito del procesador y sus posibles usos, el como se comunica el microprocesador con el JTAG y otras opciones de configuración.



2.2 Descripción de las herramientas de trabajo

En este apartado se enumerarán y explicaran las distintas herramientas software que hay que instalar para el correcto funcionamiento del entorno de trabajo.

- El primer componente software necesario es java JRE (Java Runtime Environment) [1]. Este es necesario para el funcionamiento de nuestro entorno de desarrollo que va a ser Eclipse, ya que este está basado en Java.

- El segundo componente software va a ser el que va a reunir al resto de aplicaciones software necesarias para hacer funcionar correctamente el entorno de desarrollo del microprocesador y este es Eclipse IDE [2].

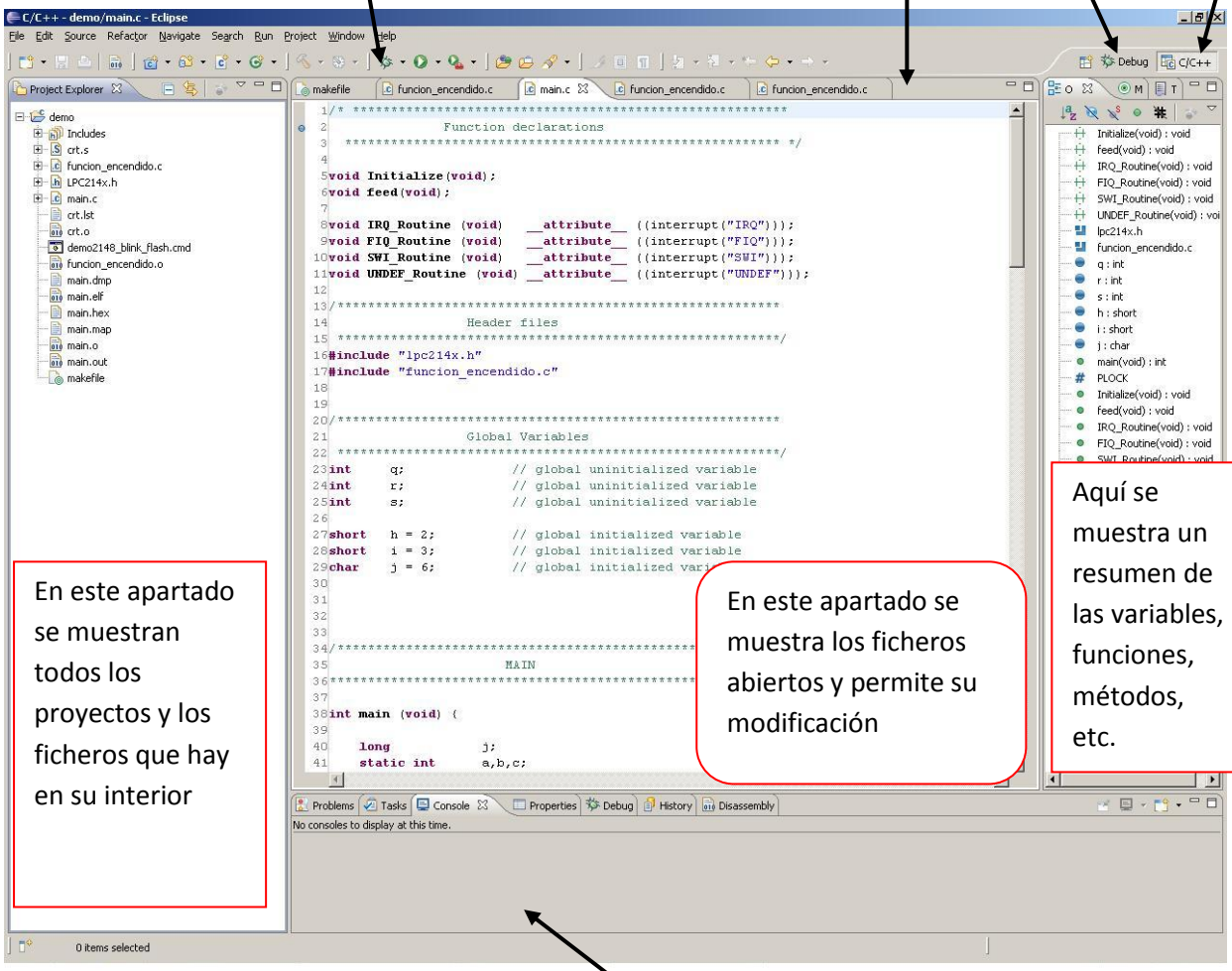
Eclipse es un entorno de desarrollo integrado gratuito, soporta distintos lenguajes de programación (en función del plugin instalado). En este proyecto como se va a utilizar el procesador lpc2148, el lenguaje que se va a utilizar es C, por lo tanto lo que nos interesa es obtener la versión que ya viene preparada para trabajar con este lenguaje, de forma que la versión de Eclipse que nosotros necesitamos es Eclipse IDE for c/c++ Developers.

Esta es una muestra de la pantalla principal de Eclipse y de algunas de sus funciones:

Si pulsamos aquí se iniciará el depurador

Por esta barra podemos irnos desplazando por los ficheros que ya están abiertos

Si pulsamos sobre debug abrimos la perspectiva de debug o si pulamos sobre c/c++ nos quedamos en la perspectiva actual



En este apartado se muestran todos los proyectos y los ficheros que hay en su interior

En este apartado se muestra los ficheros abiertos y permite su modificación

Aquí se muestra un resumen de las variables, funciones, métodos, etc.

En este apartado se muestra la información de salida del proyecto

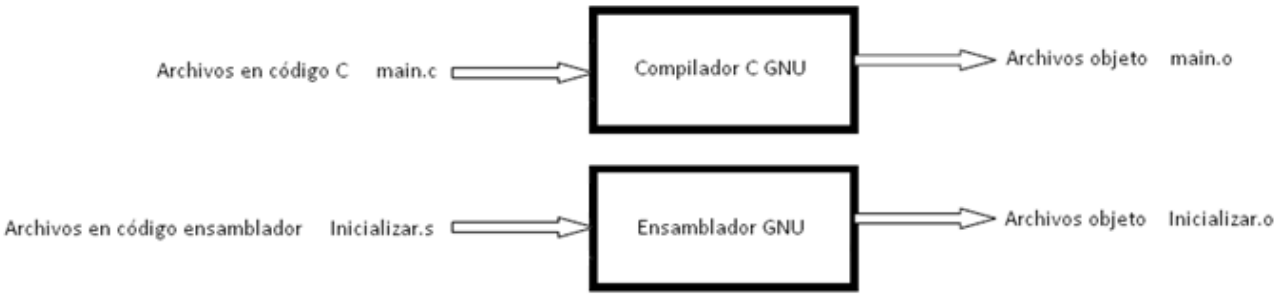
- El siguiente componente software necesario será Cygwin [3]. Cygwin es un conjunto de librerías dinámicas que se va a encargar de que podamos trabajar con el microprocesador sobre Windows, ya que el set de herramientas GNUARM [4] (más información sobre ella en el siguiente punto) fue desarrollado para sistemas Unix y por lo tanto no es apto para Windows.

  De forma que la función de Cygwin es la de engañar al set de herramientas GNU y hacerle creer que está trabajando sobre Unix. Para poder hacerlo utiliza dos partes diferenciadas. La primera es cygwin1.dll que actúa como capa de emulación del API de Linux dotando de funcionalidad al API de Linux en Windows, y la otra parte está compuesta de un conjunto de herramientas que simulan el entorno de trabajo de Linux.

- La próxima herramienta a utilizar es GNUARM [4]. Es la herramienta que va a compilar, enlazar y depurar los programas escritos en C/C++.

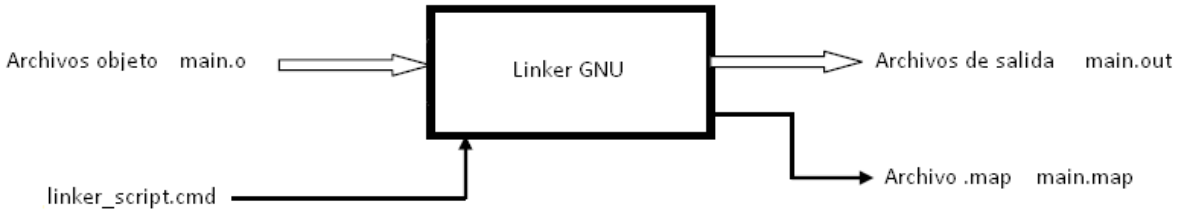
  El compilador GNU C [5] se caracteriza por su integridad y la amplia gama de generadores de código que posee dirigidos a la mayoría de arquitecturas de los microprocesadores más utilizados. Además del compilador; GNUARM incluye un ensamblador, enlazador (linker), depurador y bibliotecas entre otras utilidades. Este compilador de C de GNUARM proporciona una velocidad y un rendimiento del código un poco inferior al proporcionado por el compilador de ARM.

  La función que el compilador va a realizar en nuestro caso es la de traducir los programas escritos en C a lenguaje máquina. El ensamblador realiza una tarea similar, pero en vez de traducir a lenguaje máquina programas escritos en C, va a convertir a lenguaje máquina los programas escritos en ensamblador. Pero tanto la salida del compilador, como la salida del ensamblador, no es del todo el lenguaje máquina que se va a ejecutar en el LPC2148, ya que hay que añadir las direcciones de memoria en las que van alojadas las distintas instrucciones en el microprocesador. Y esto se soluciona más adelante con el linker [6], que nos va permitir alojar el programa en cualquier dirección de memoria.

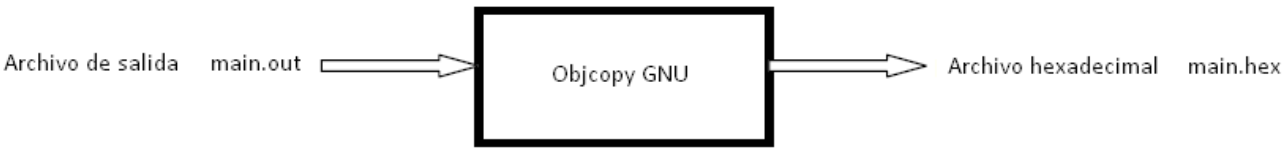


El linker GNU, se va a encargar de unir los archivos objeto que se han creado con el compilador y el ensamblador mediante el script con extensión .cmd (linker_script.cmd [7]) se crea un único fichero con extensión .out.

El fichero .out generado contiene el lenguaje máquina que se ejecuta en la memoria RAM del microprocesador y la información relativa al depurador. El linker, también produce un fichero de información con extensión .map que indica la longitud de los módulos, que posición ocupan en la memoria del microprocesador, a que sección de código pertenecen, etc.



Otra de las herramientas GNU necesarias si queremos cargar el programa en la memoria flash es la herramienta Objcopy [8]. Esta herramienta se va a encargar de convertir el fichero main.out creado anteriormente en otro fichero con extensión .hex que va a ser el que se va a cargar en la memoria flash del microprocesador.



La llamada a las herramientas de GNUARM se puede realizar mediante línea de comandos e irlos introduciendo en el orden descrito anteriormente (primero compilar, segundo ensamblar y tercero unir). Pero resulta mucho mas sencillo utilizar la herramienta Make que incorpora Cygwin, esta herramienta busca en el proyecto el fichero makefile [9] (el cual previamente se ha escrito), donde viene indicado el uso de las herramientas GNU en el orden correcto junto a los flags de configuración de las herramientas. La llamada a la herramienta Make se realiza a través de Eclipse.

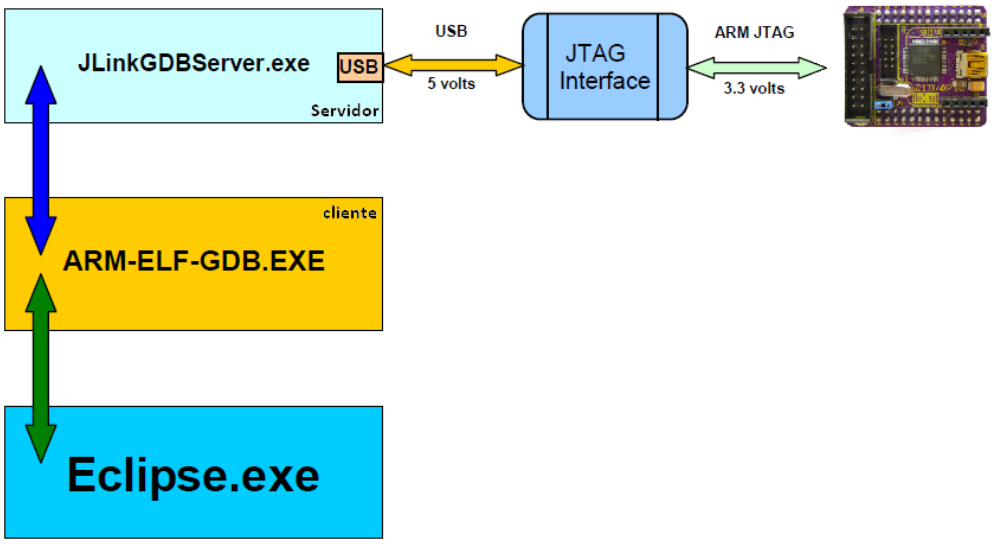
También el depurador de GNUARM está integrado con Eclipse, de forma que desde él se puede realizar la depuración de los programas cargados en la memoria flash del microprocesador con el uso de breakpoints, ejecución paso a paso y el manejo en tiempo real del estado de las variables en el microprocesador.

- Finalmente, la última herramienta necesaria para el completo funcionamiento del entorno de trabajo es el software del JTAG. Este software es el que se va a encargar de que el PC reconozca al JTAG y que a través de él podamos acceder al microprocesador. El software proporcionado con el JTAG tiene varias herramientas, pero las que vamos a utilizar en este manual son “J-Flash ARM” [10] y “J-Link GDB Server” [11].

  La primera de ellas será la encargada de conectar al ordenador a través del puesto USB con el procesador mediante el JTAG para cargar en la memoria flash el código ejecutable (archivo *.hex) y por último dar la orden de comienzo de la aplicación.

  La segunda aplicación será necesaria a la hora de utilizar el depurador, pues es el intermediador entre eclipse y el microprocesador. Al iniciar el depurador con el eclipse, se abre una herramienta auxiliar llamada arm-elf-gdb.exe. Esta herramienta lee el contenido de main.out y manda instrucciones al puerto TCP 2331 y en este momento es cuando entra en funcionamiento nuestra herramienta “J-Link GDB Server”. Ambos (arm-elf-gdb.exe y “J-Link GDB Server”) van a trabajar en una

relación cliente servidor, arm-elf-gdb.exe va a ser el cliente que pide información sobre el LPC2148, en función del contenido de main.out y "J-Link GDB Server" va a actuar como servidor interrogando al LPC2148 vía JTAG aportando la información necesaria al cliente.



3. Descarga de las herramientas de trabajo

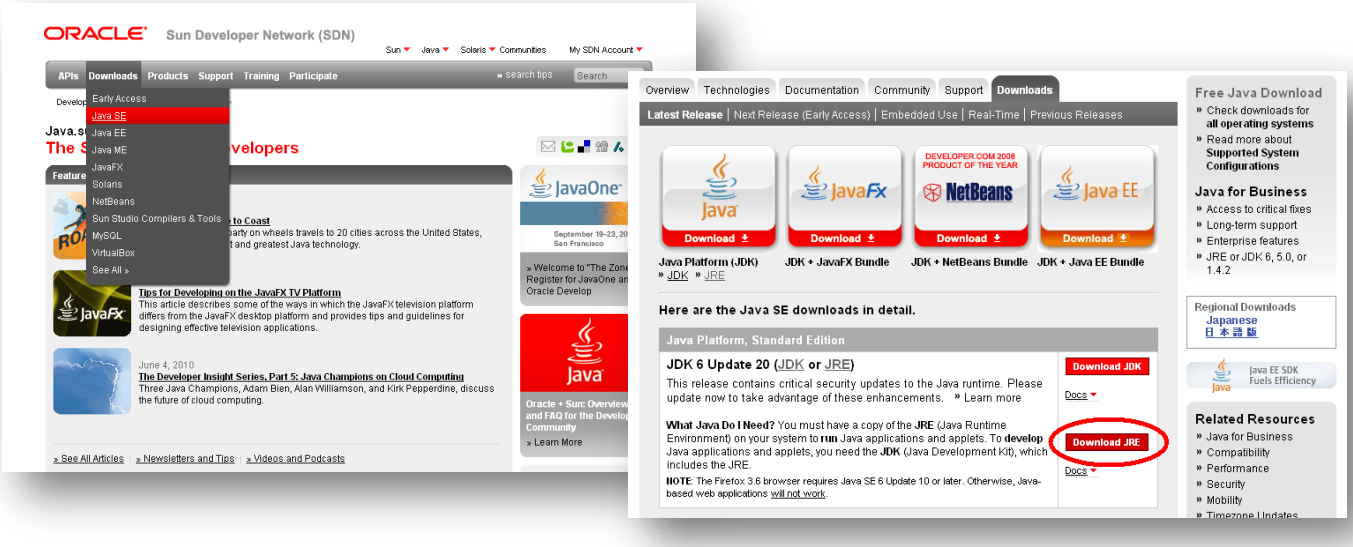
Componentes necesarios:

Los componentes necesarios son los mencionados en el apartado anterior:

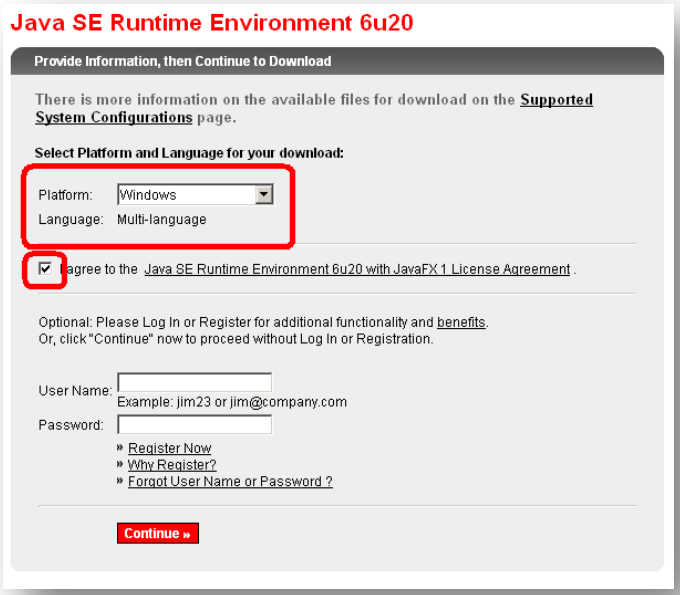
1. SUN Java Runtime.[1]
2. Eclipse IDE for c/c++ Developers.[2]
3. Cygwin.[3]
4. GNUARM.[4]
5. J-Link ARM [12]

3.1 Descarga de Java (JRE)

Dirigirse a la página: http://java.sun.com/ [1] y descargarse la última versión disponible de java. El Java necesario para que Eclipse pueda ejecutarse es el JRE.

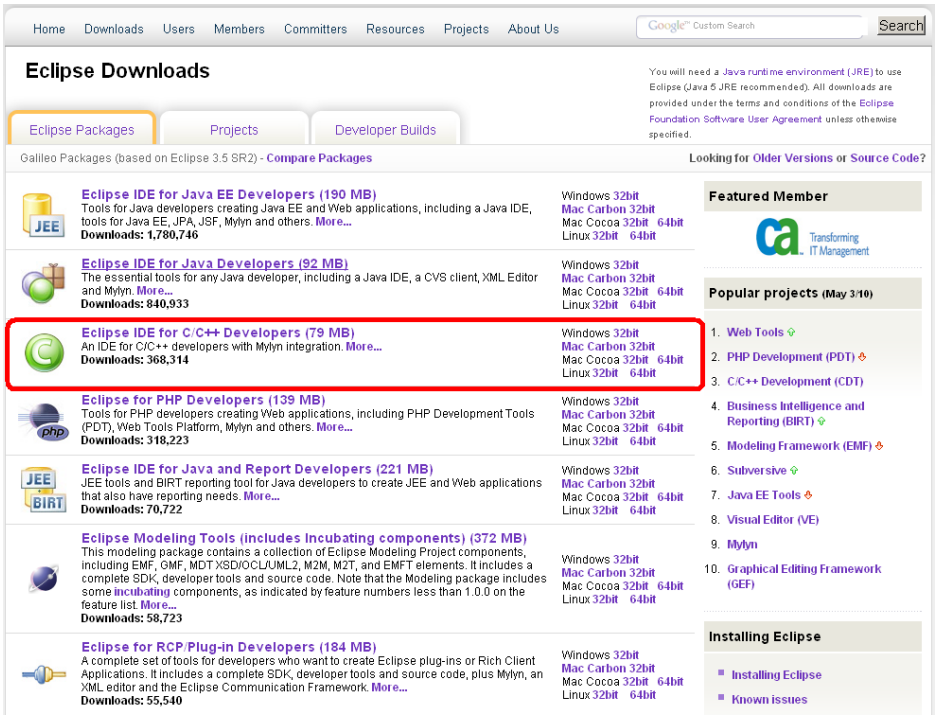


Se selecciona el sistema operativo y se acepta la licencia.Pulsar sobre "***Continue"*** y descargar el ejecutable.

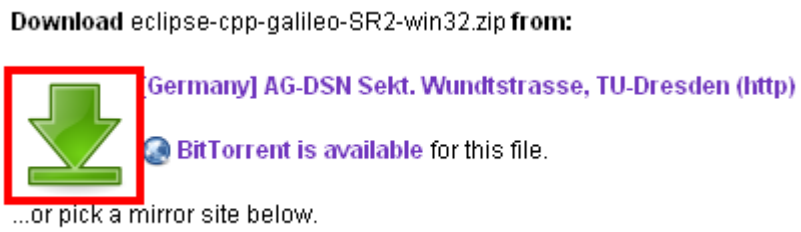


3.2 Descarga de Eclipse:

Para ello, hay que dirigirse a la página: http://www.eclipse.org/ [2] y pulsar en "***Download"***, se llegará a esta página y se seleccionará la opción marcada.

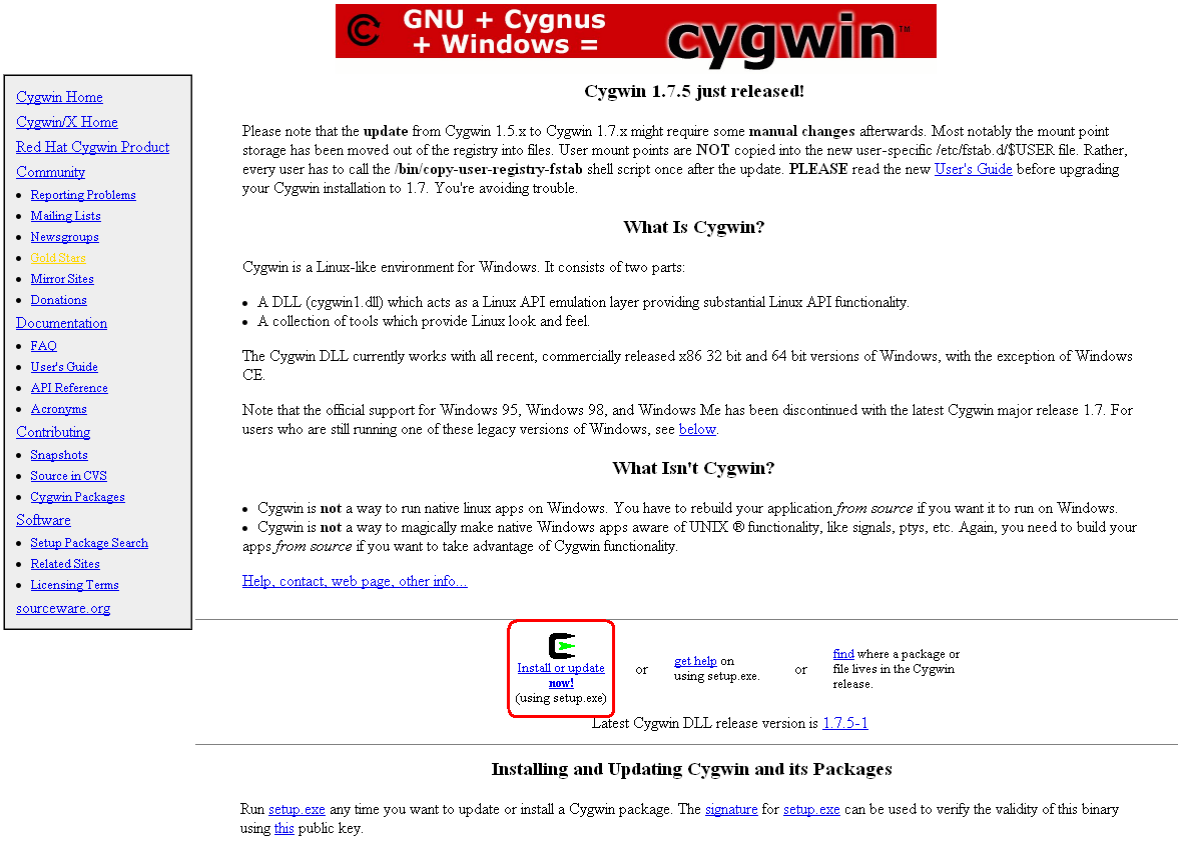


Pulsar en la flecha e inmediatamente comenzará la descarga.



3.3 Descarga Cygwin:

Acceder a la página: http://www.cygwin.com/ [3] y pulsar sobre **"Install or Update".**



3.4 Descarga GNUARM

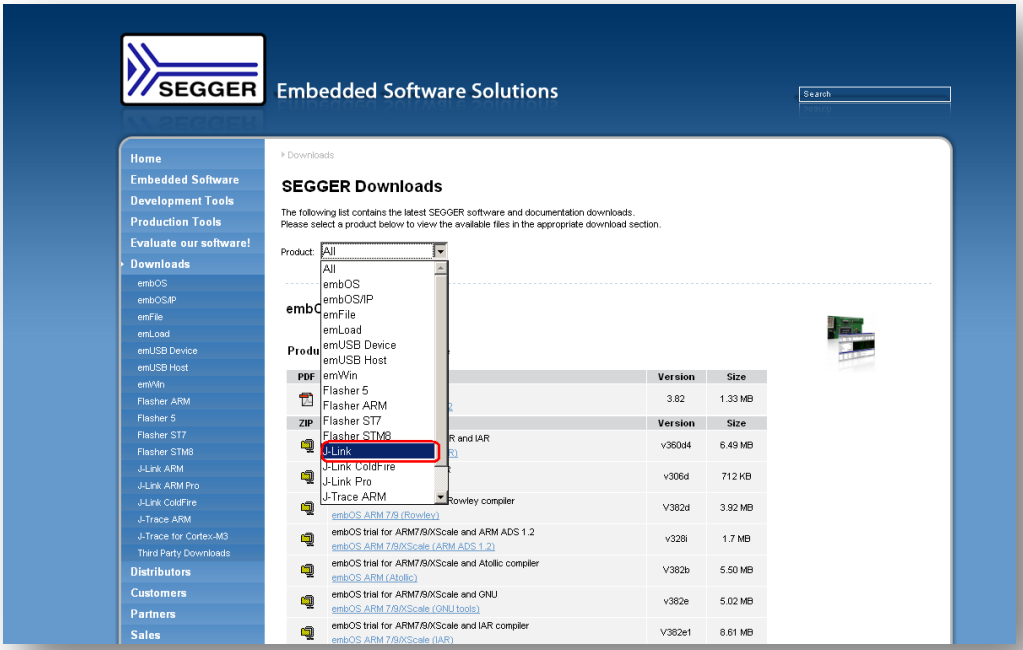
Acceder a la página: http://www.gnuarm.com/ [4]

Pulsar sobre Files y dirigirse a la sección de "***Binaries***" y descargarse la última versión disponible.

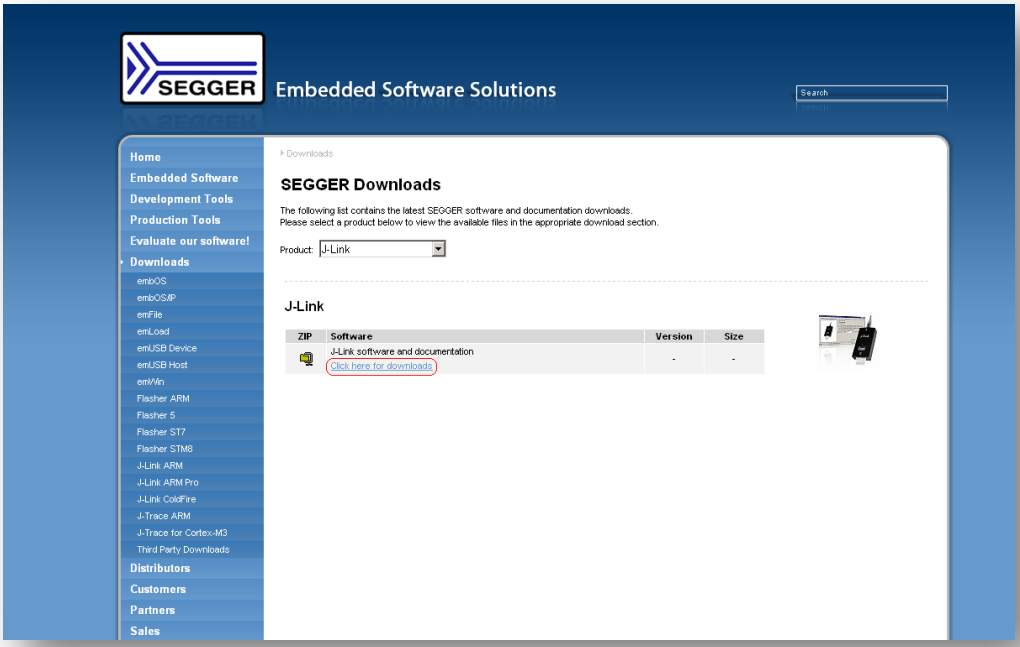
3.5 Descarga de software de SEGGER:

Ir a la dirección www.segger.com [12], y dirigirse a la sección de "***Downloads***".

Ahora buscar en el menú desplegable *"**J-Link**"* y pulsar sobre él.



Pulsar en "***Click here for downloads***"

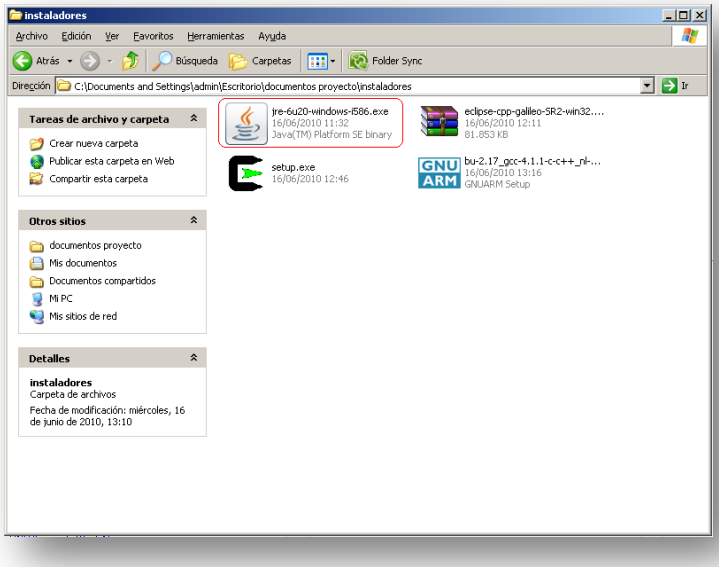


Y en la siguiente ventana que aparece seleccionar "**Software and documentation pack V4.14d**" momento en el cual comenzará la descarga.

4. Instalación de las herramientas de trabajo

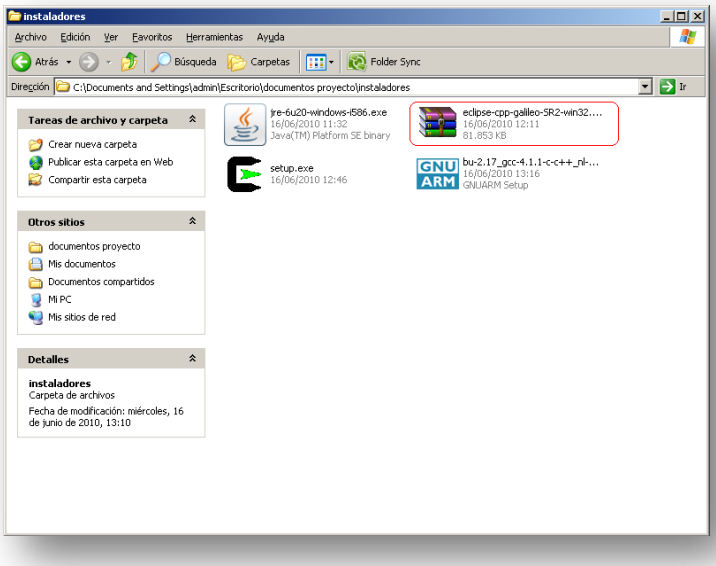
4.1 Instalación de Java:

Dirigirse al directorio donde se han descargado todos los instaladores y ejecutar el instalador de Java. Instalar en la carpeta indicada por defecto.



4.2 Instalación de Eclipse:

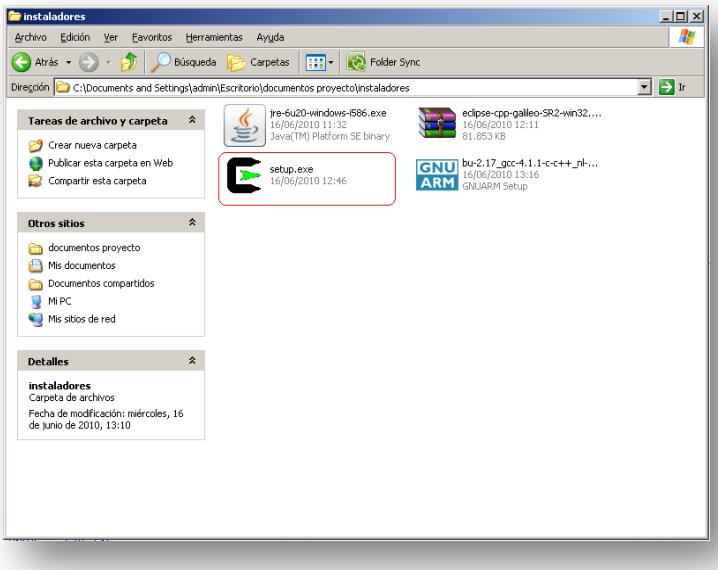
Dirigirse al directorio donde se han descargado todos los instaladores y descomprimir el fichero *.zip de Eclipse en la carpeta donde se desee alojar Eclipse. Una vez realizado, la instalación de eclipse ha finalizado.



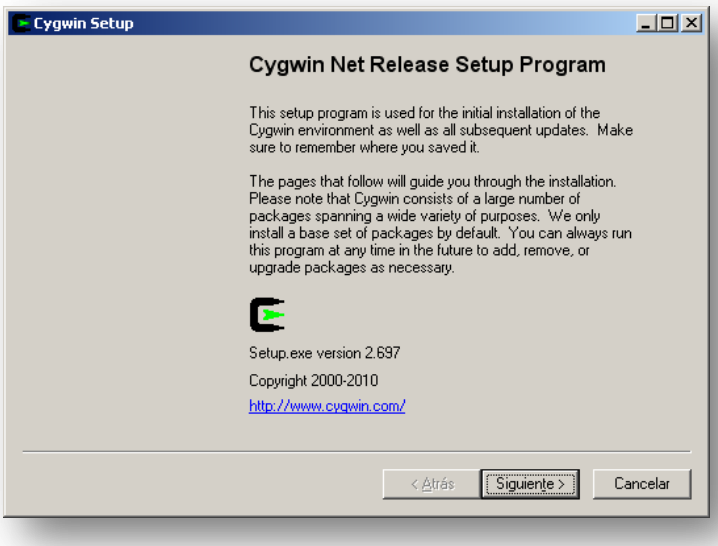
4.3 Instalación de Cygwin [13] [22]:

4.3.a Instalación de Cygwin mediante internet:

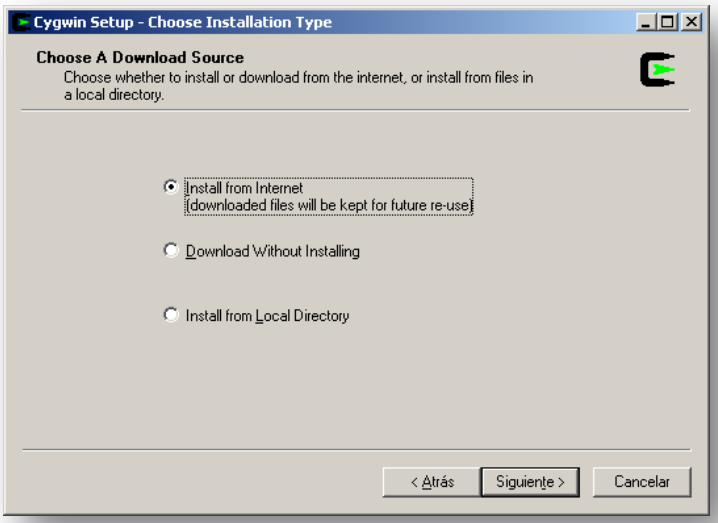
Abrir el directorio donde se han descargado todos los instaladores y ejecutar el instalador de Cygwin.



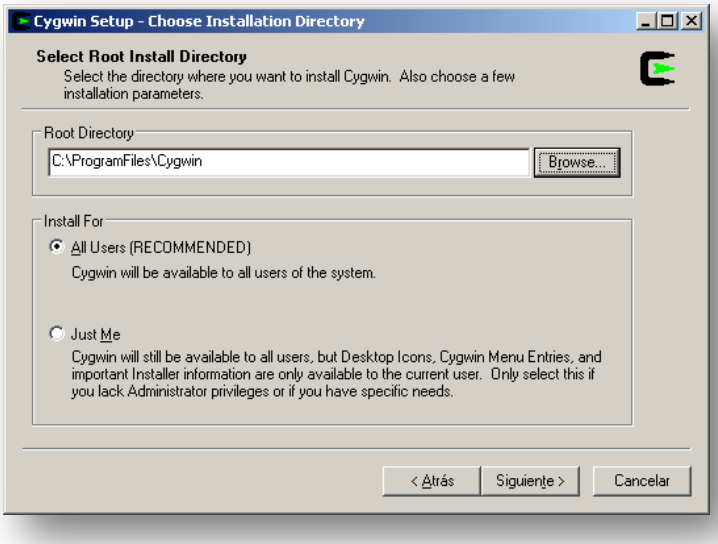
Pulsar en “***Siguiente”***



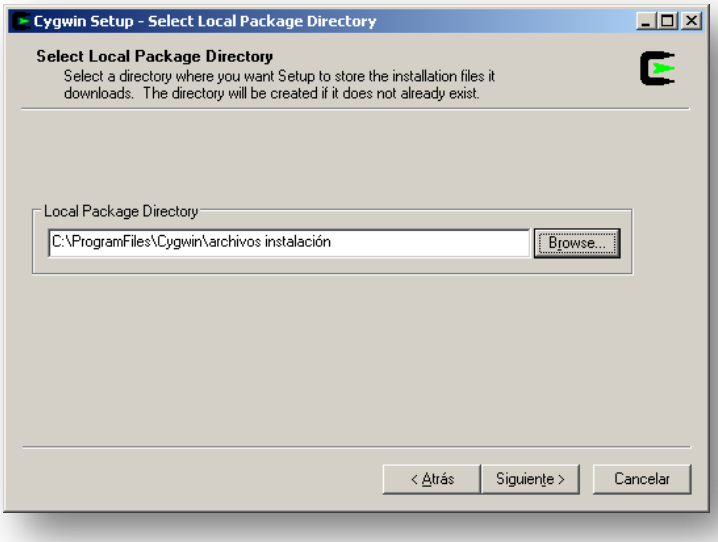
En este caso, como instalación se está realizando a través de Internet, seleccionar la opción de instalar desde Internet y pulsar "***Siguiente***".



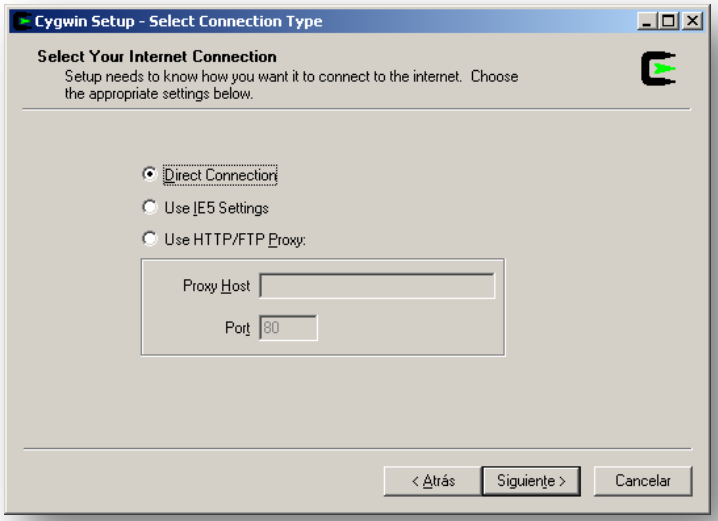
Seleccionar la carpeta en la cual se quiere instalar Cygwin. Ha de ser una carpeta cuyo nombre no contenga espacios. Una buena opción es c:\cygwin. Pulsar "***Siguiente***".



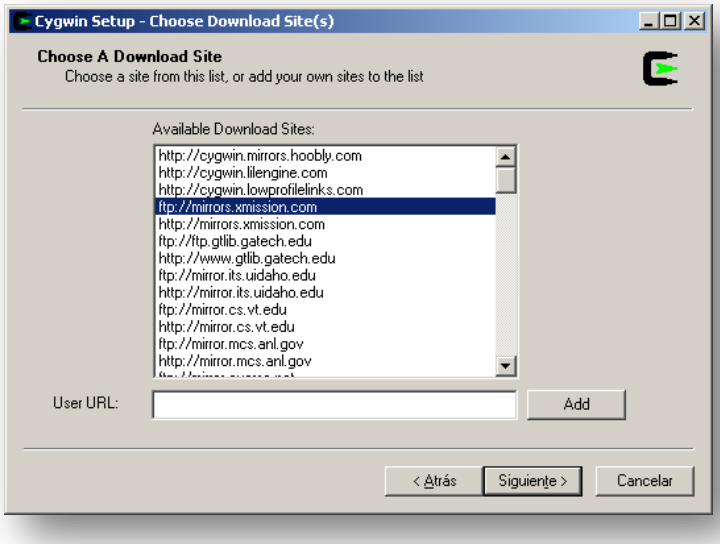
En el siguiente paso, seleccionar la carpeta donde se quiere que se almacenen los archivos descargados para la instalación de Cygwin.



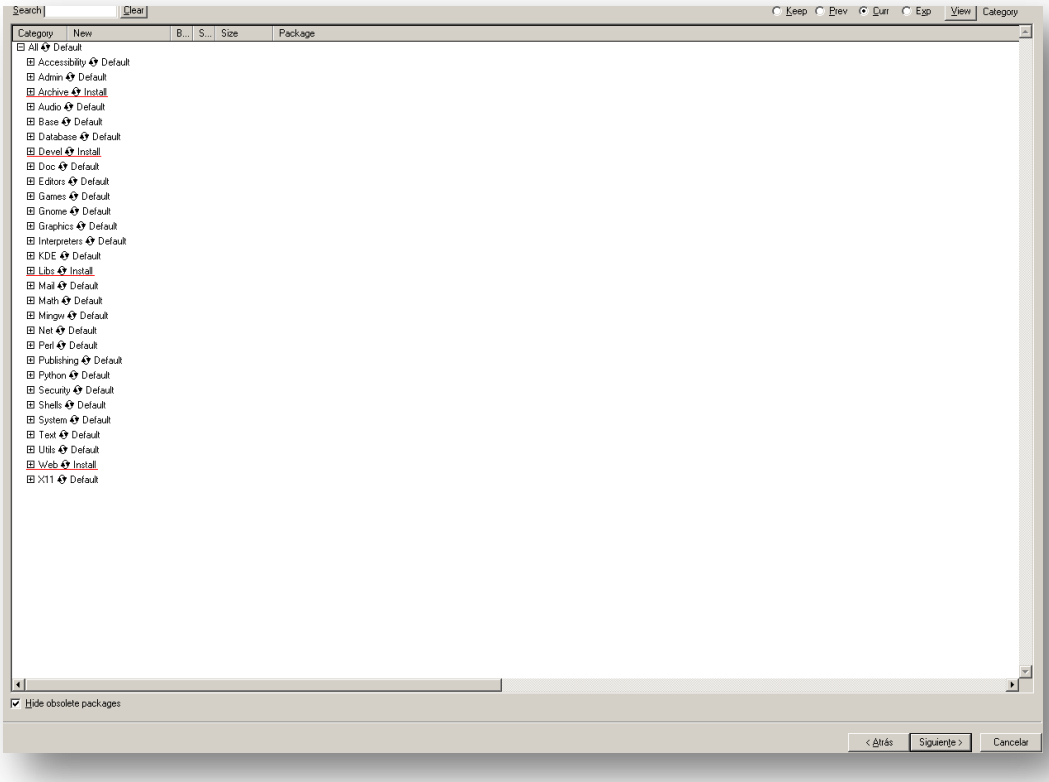
En la siguiente ventana, seleccionar "***Direct Connection***".



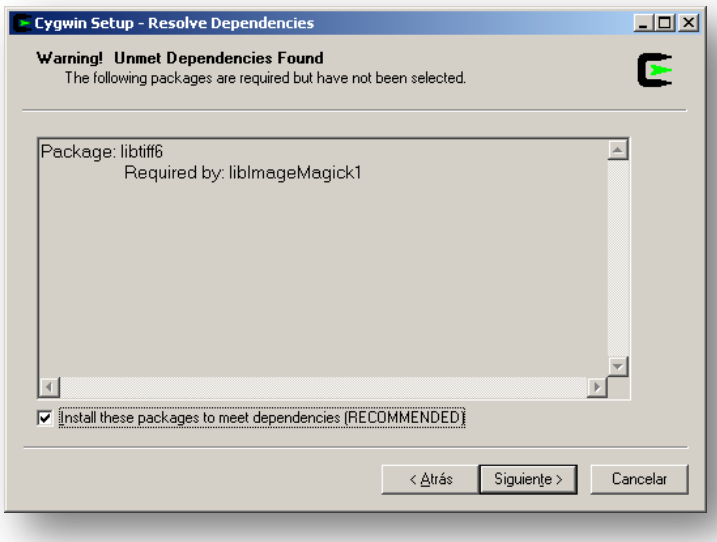
Ahora seleccionar el servidor de descarga:



Y tras ello pulsar "**Siguiente"** y seleccionar los paquetes que hacen falta para que GNU funcione correctamente sobre Windows. Estos paquetes son "**Archive"**, "**Devel"**, "**Libs"** y "**Web"**. Para seleccionarlos, hacer click sobre hasta que la opción cambie de **"Default"** a **"Install"** (Puede tardar un poco desde que se pulse hasta que cambie de estado).

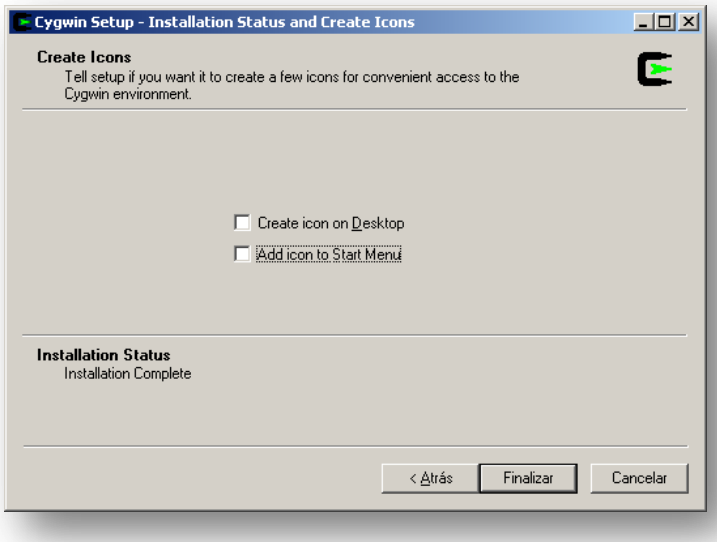


Pulsar sobre **“Siguiente”** y en el caso de que se indique que hay algún paquete que es necesario y no se ha seleccionado, Indicar que se instale y pulsar **“Siguiente”**.

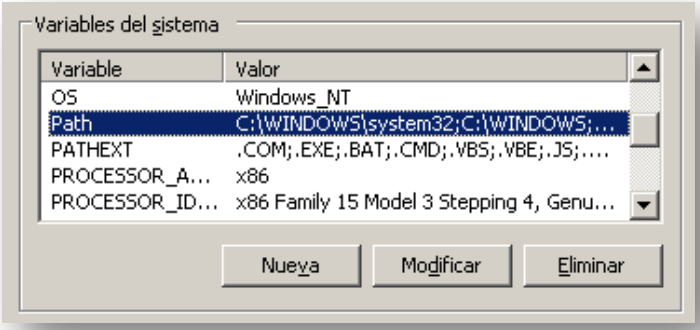


Ahora, se comenzarán a descargar los paquetes seleccionados, estos tienen alrededor de 600Mb por lo que el tiempo de descarga puede superar la hora si el servidor seleccionado no nos ofrece una buena velocidad de descarga.

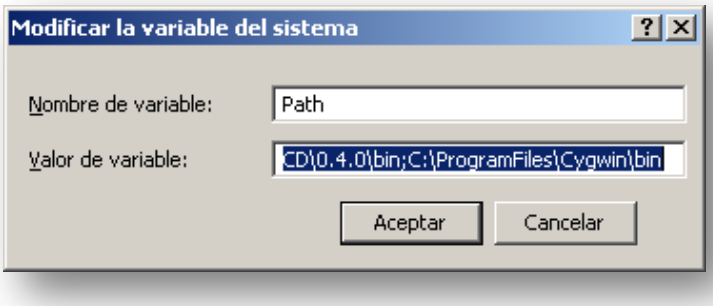
Una vez finalizada la descarga, se instalarán los paquetes y una vez terminado, pulsar sobre **“Finalizar”** y la instalación habrá terminado.



Ahora hay que añadir el directorio …\cygwin\bin en la Path, para ello, pulsar ***"Inicio -> Panel de control -> Sistema -> Opciones avanzadas -> Variables de entorno"*** y en variables del sistema buscar la variable Path:



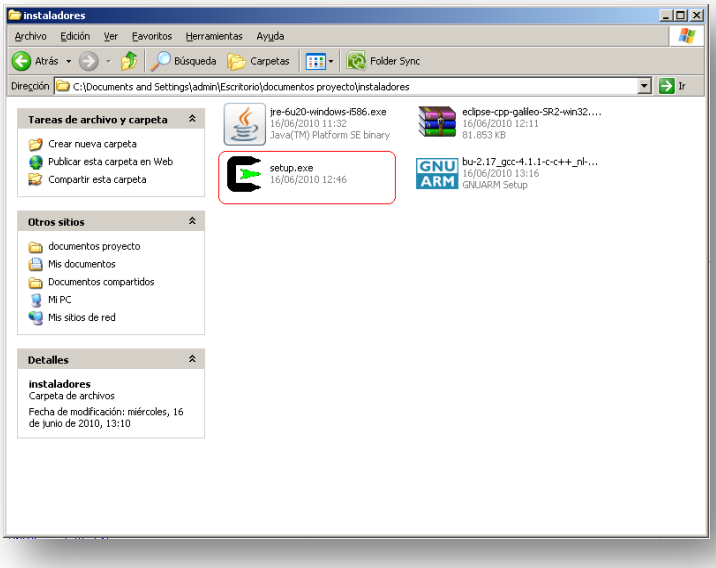
Se selecciona y se pulsa sobre "**modificar"** para añadir la dirección y añadir la siguiente ruta: ";C:\ProgramFiles\Cygwin\bin" (no incluir las comillas). De forma que queda:



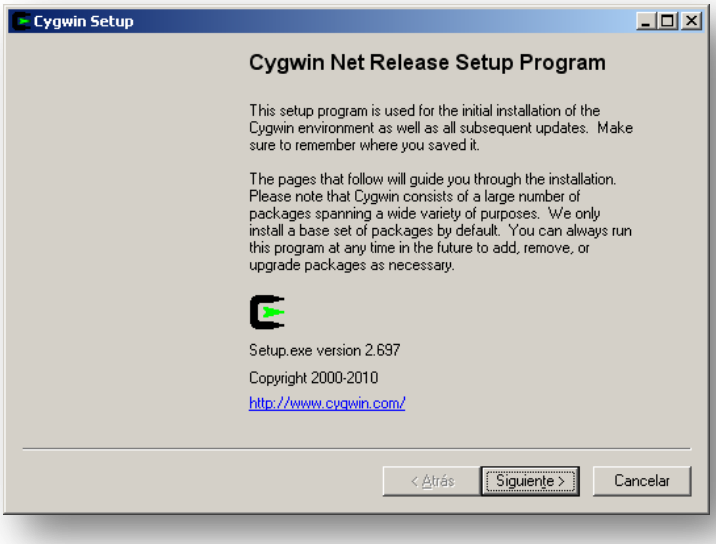
Pulsamos sobre aceptar y la instalación de Cygwin queda finalizada. Ir al apartado **4.4**

4.3.b Instalación de Cygwin sin Internet:

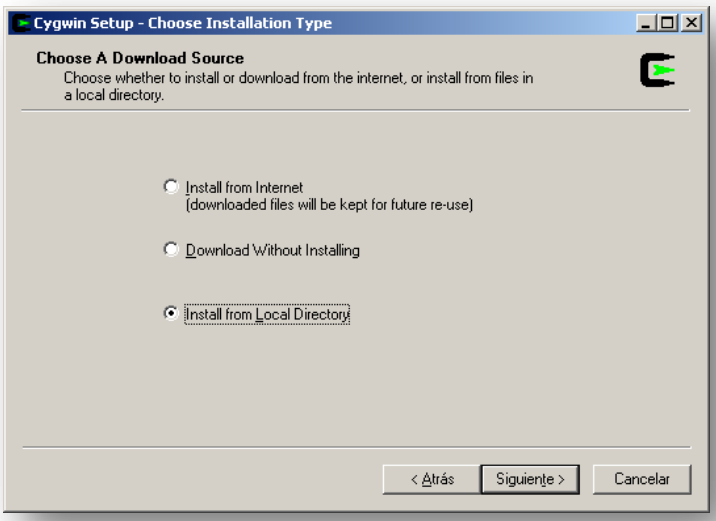
Abrir el directorio donde se han descargado todos los instaladores y ejecutar el instalador de Cygwin.



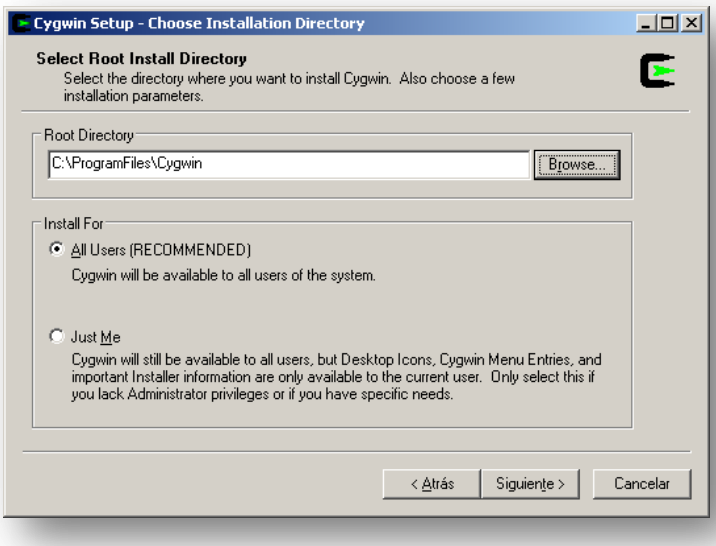
Hacer click en ***"Siguiente"***



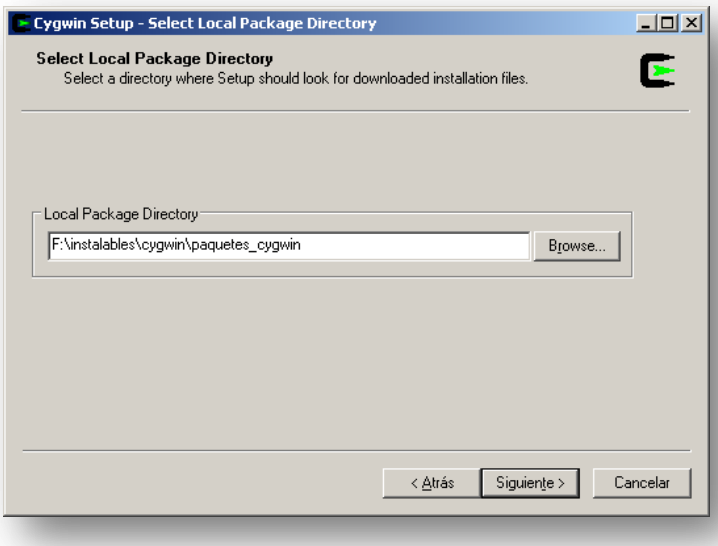
En este caso, como queremos hacer la instalación sin Internet, seleccionamos la opción "***Install from local directory***".



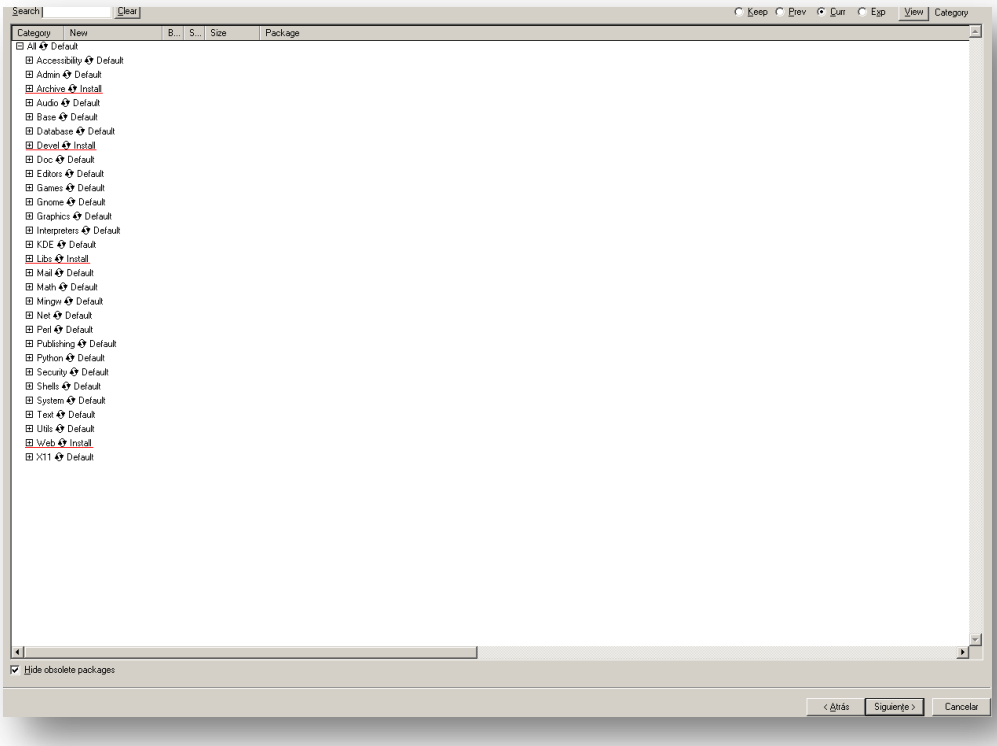
Seleccionar la carpeta en la cual se quiere instalar Cygwin. Ha de ser una carpeta cuyo nombre no contenga espacios. Una buena opción es c:\cygwin. Pulsar "***Siguiente***".



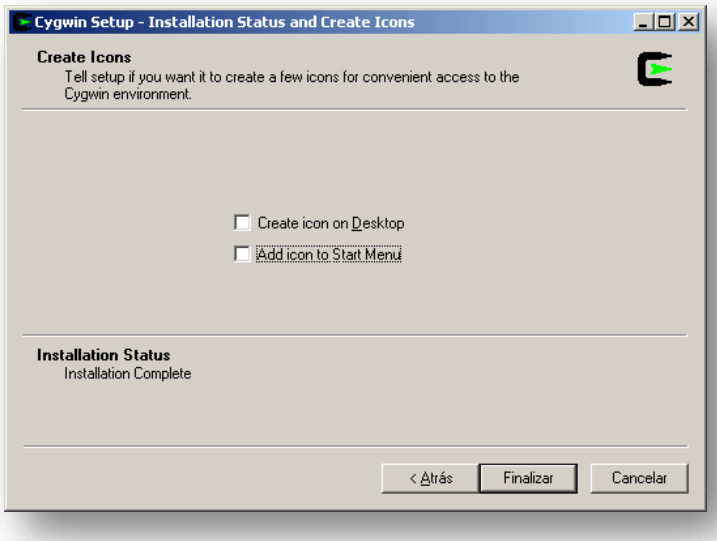
En el siguiente paso, seleccionar la carpeta de donde queremos extraer los paquetes necesarios para la instalación de Cygwin.



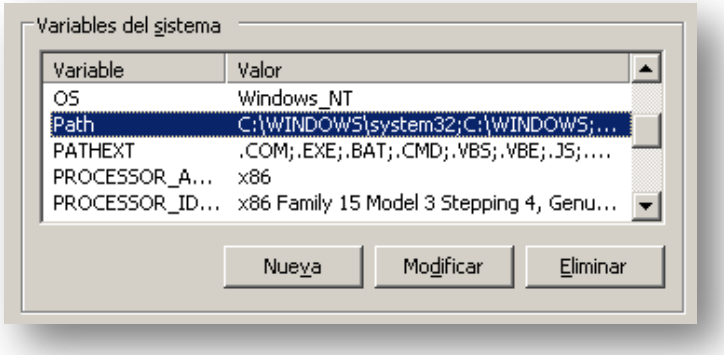
Pulsar "***siguiente"*** y seleccionar los paquetes que hacen falta para que GNU funcione correctamente sobre Windows. Estos paquetes son "**Archive"**, "**Devel"**, "**Libs"** y "**Web"**. Para seleccionarlos, hacemos click sobre hasta que la opción cambie de **"Default"** a **"Install"** (Puede tardar un poco desde que se selecciona hasta que cambia de estado).



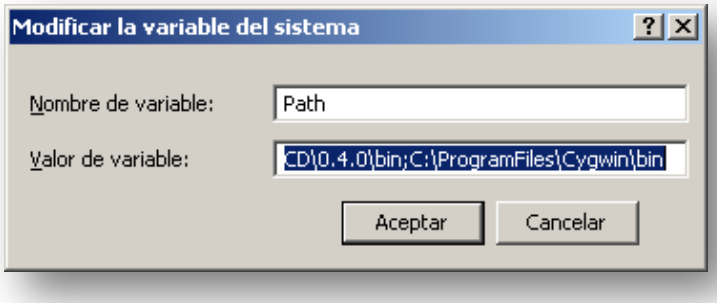
Una vez seleccionados los paquetes, se instalarán y una vez terminado, pulsar sobre "**finalizar"** y la instalación habrá terminado.



Ahora hay que añadir el directorio …\cygwin\bin en la Path, para ello, pulsar "***Inicio -> Panel de control -> Sistema -> Opciones avanzadas -> Variables de entorno"*** y en variables del sistema buscar la variable Path:



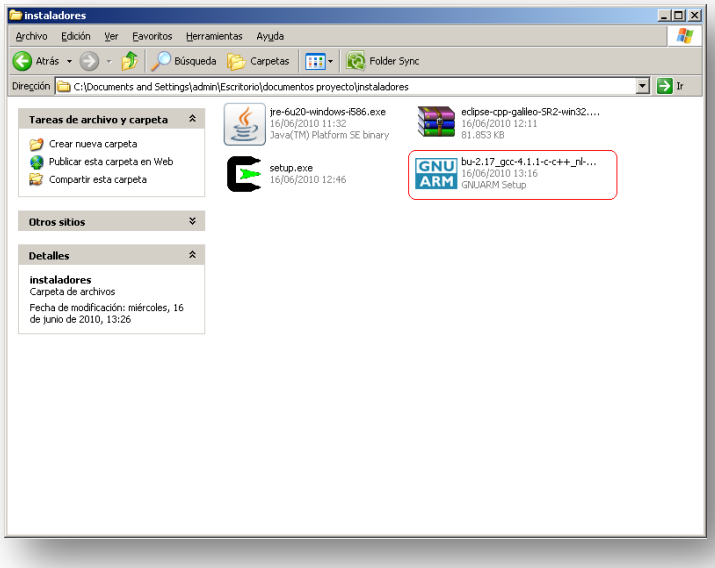
Se selecciona y se pulsa sobre "**modificar"** para añadir la dirección y añadir la siguiente ruta: ";C:\ProgramFiles\Cygwin\bin" (no incluir las comillas). De forma que queda:



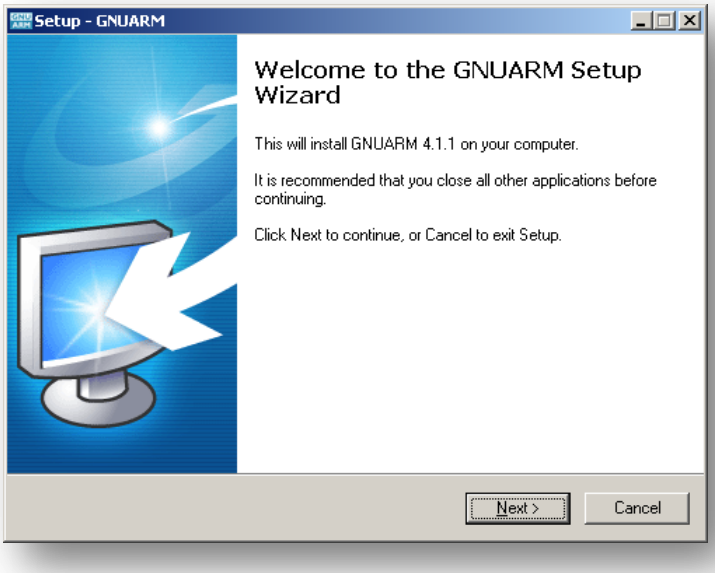
Pulsar sobre **"Aceptar"** y la instalación de Cygwin queda finalizada.

4.4 Instalación de GNUARM:

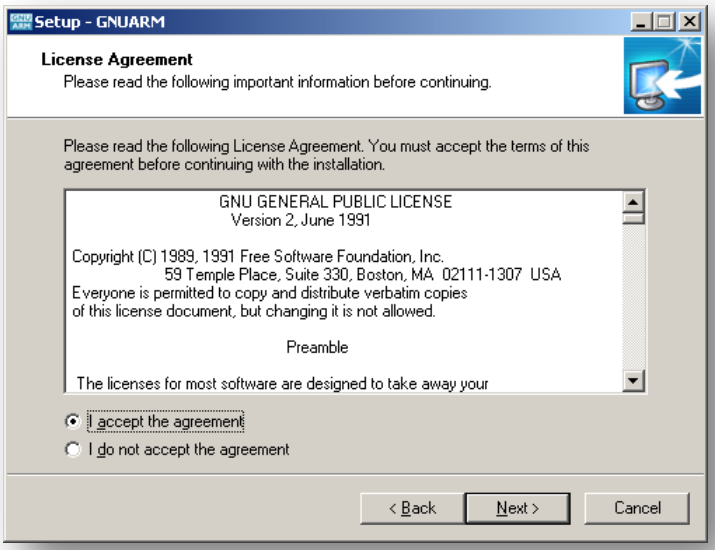
Abrir el instalador de GNUARM



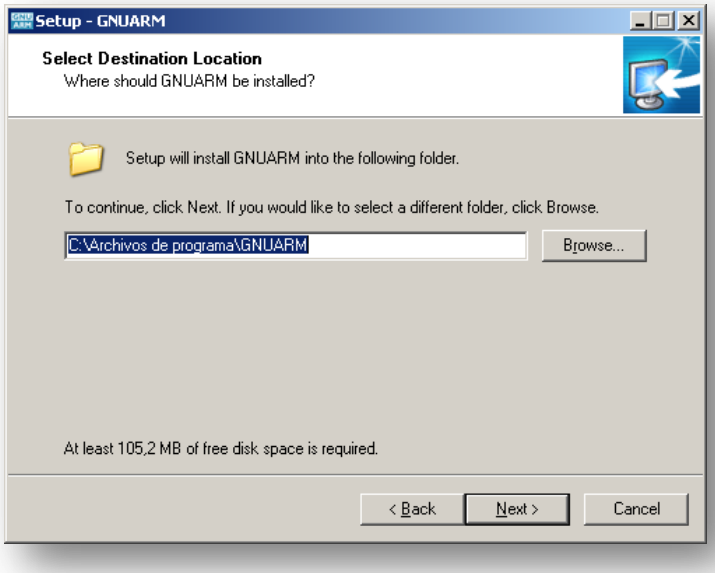
Hacer click sobre "***Next".***



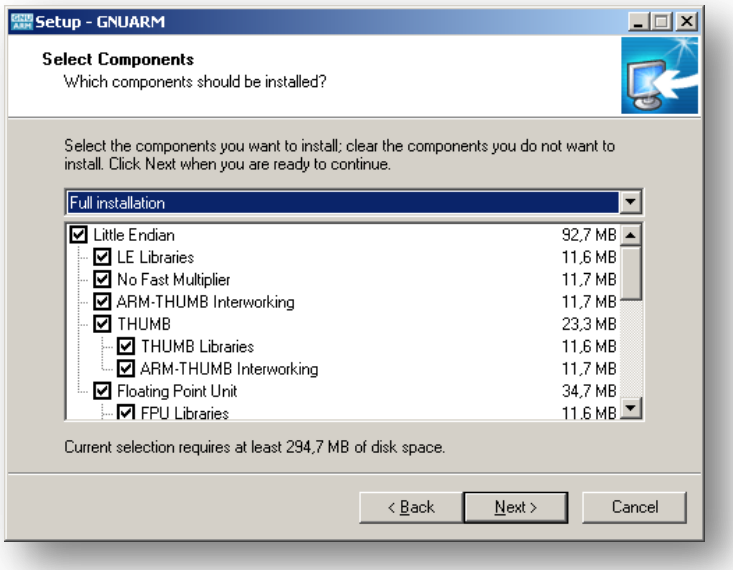
Se acepta la licencia



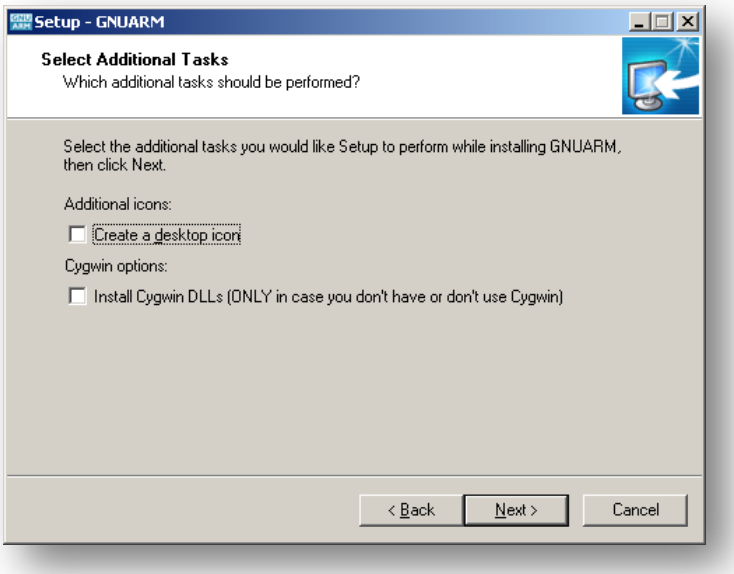
Seleccionar la carpeta en la cual se quiere instalar GNUARM



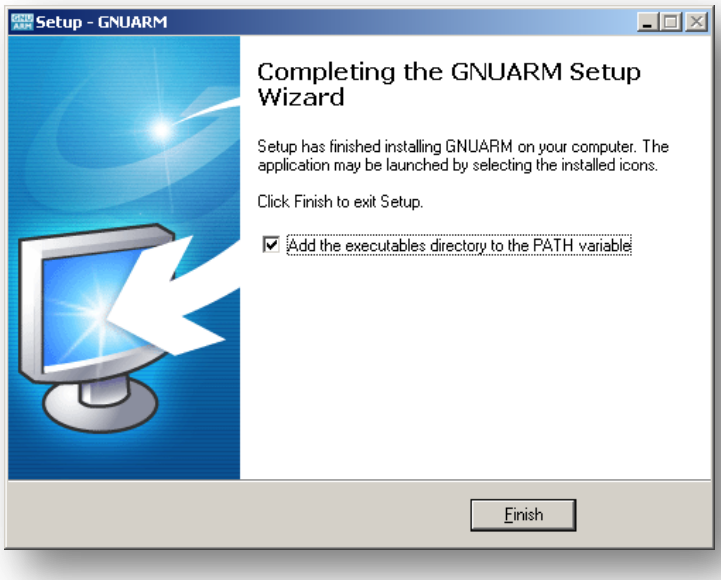
Seleccionar "***Full Installation".***



Se sigue sobre "**Next"** hasta llegar a la pantalla mostrada en la imagen inferior, donde hay que dejar sin marcar las dos opciones, (la primera de ellas es para crear el acceso directo de GNUARM y no es necesario y la segunda opción es para instalar las librerías de Cygwin pero no es necesario pues ya se han instalado anteriormente).

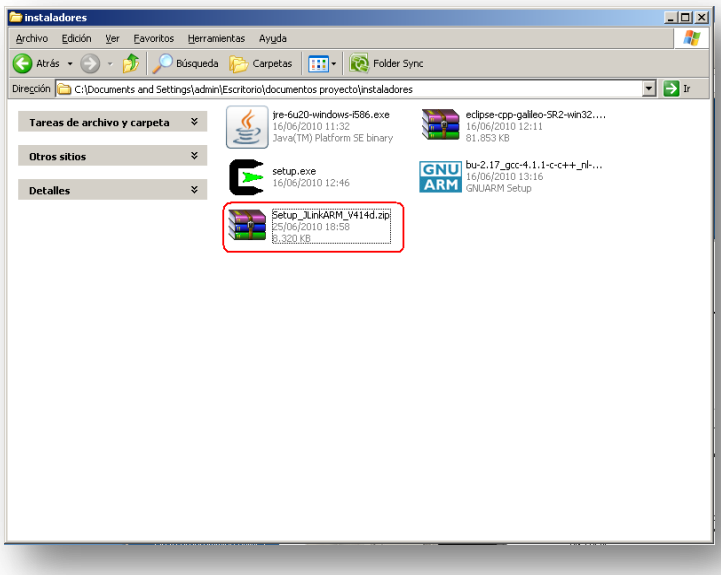


Hacer click sobre **"Next"** y en la siguiente pantalla sobre **"Install"**. Una vez finalizada la instalación, añadir las variables en la PATH de Windows.

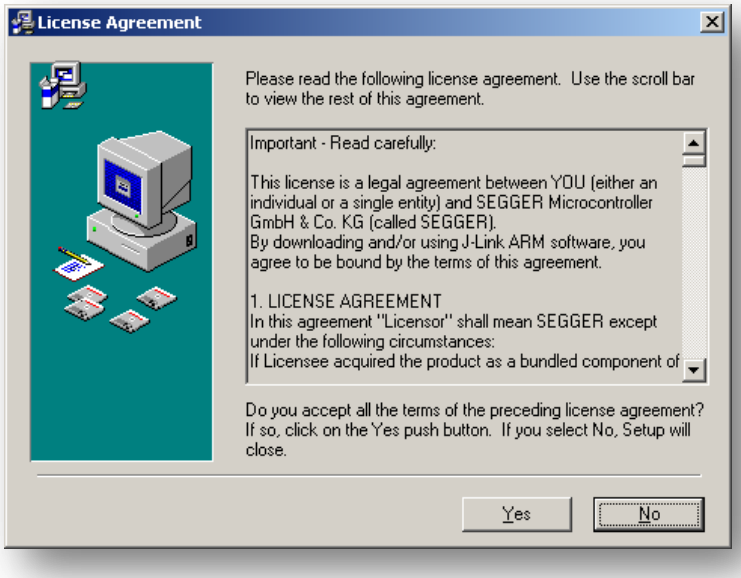


4.5 Instalación de J-link:

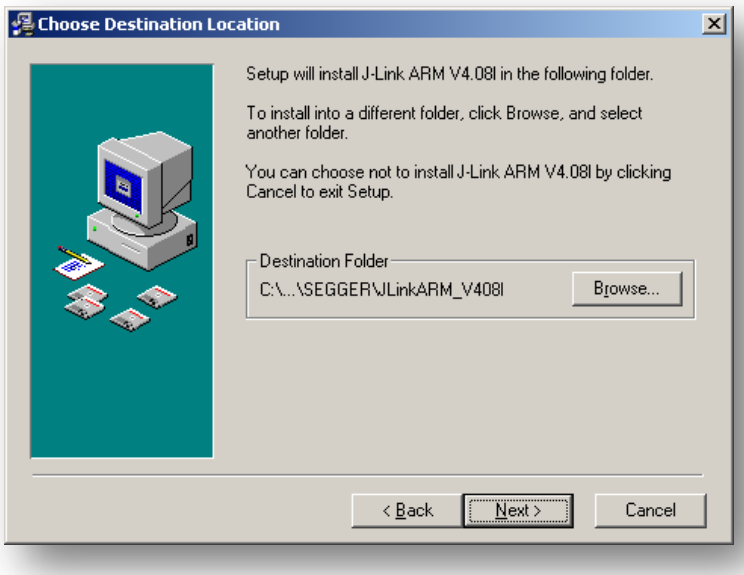
Abrir el instalador de J-Link



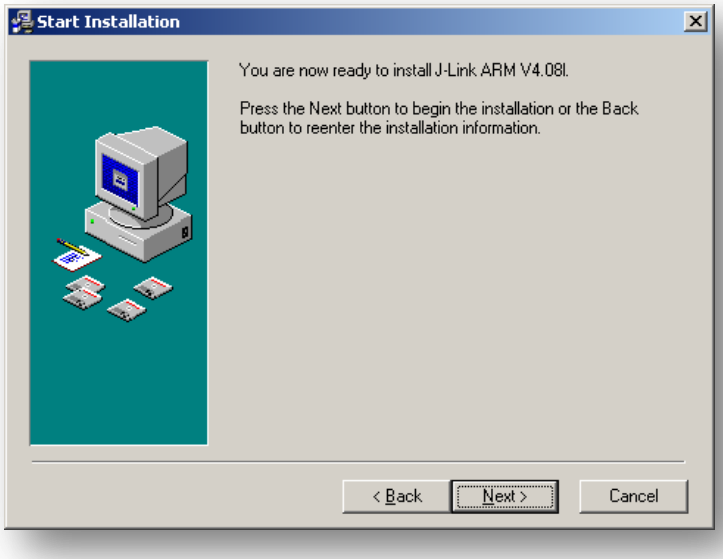
Aceptar la licencia y continuar con la instalación.



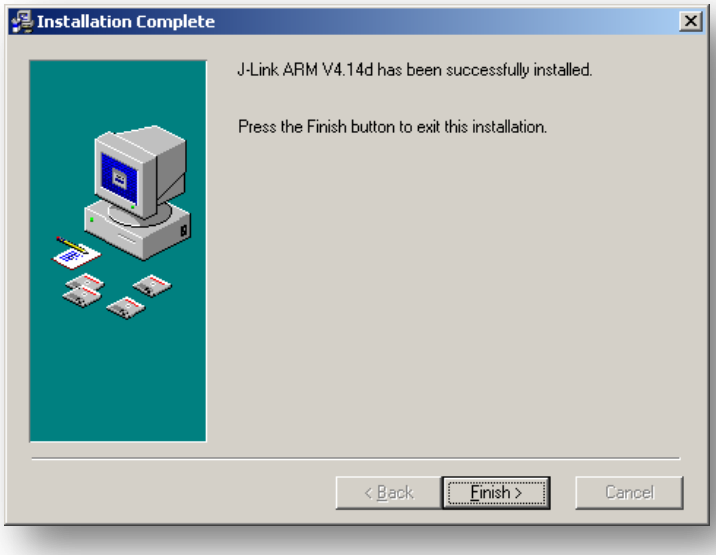
Continuar pulsando sobre "***Next***" hasta llegar a la pantalla mostrada en la imagen inferior donde se ha de seleccionar el directorio en el que se quiere instalar J-link. Por defecto, **"C:\ Archivos de Programa\SEGGER\...."**.



En esta pantalla pulsar sobre "***Next***" y comienza la instalación.

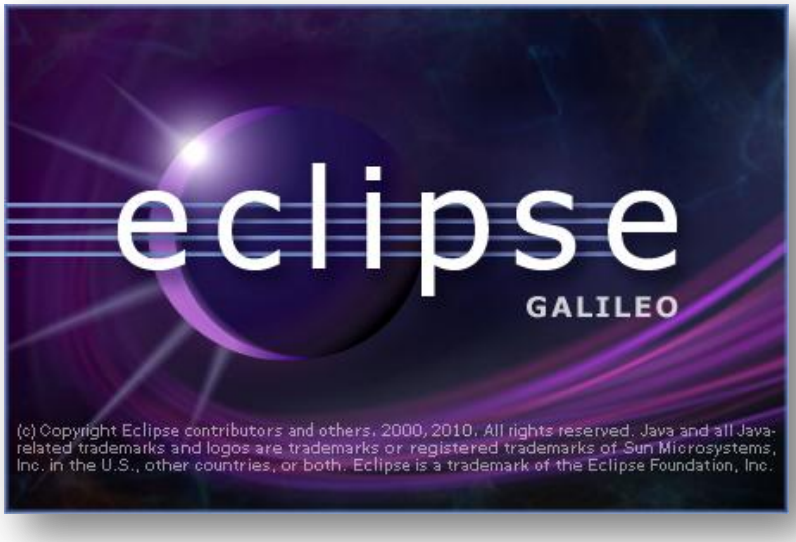


Una vez finalizada la instalación, la siguiente pantalla nos indicará que el programa ha sido instalado correctamente y pulsamos sobre "***Finish***" para terminar.

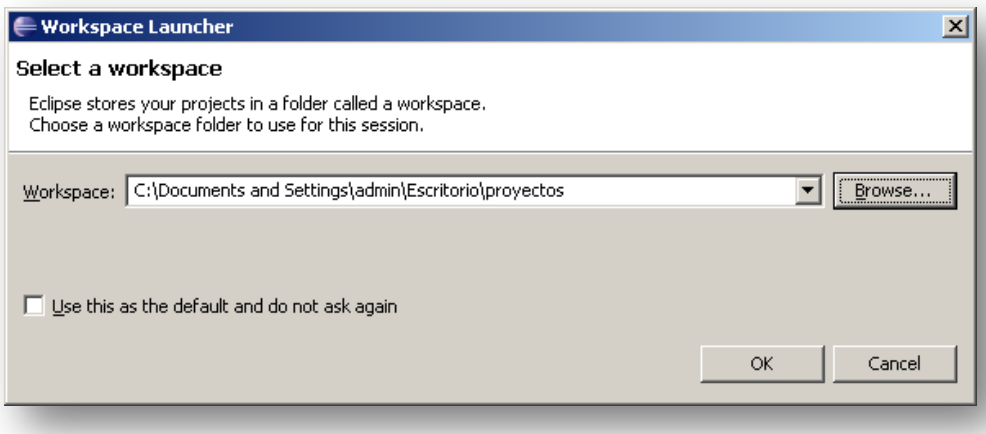


5. Configuración del entorno de trabajo

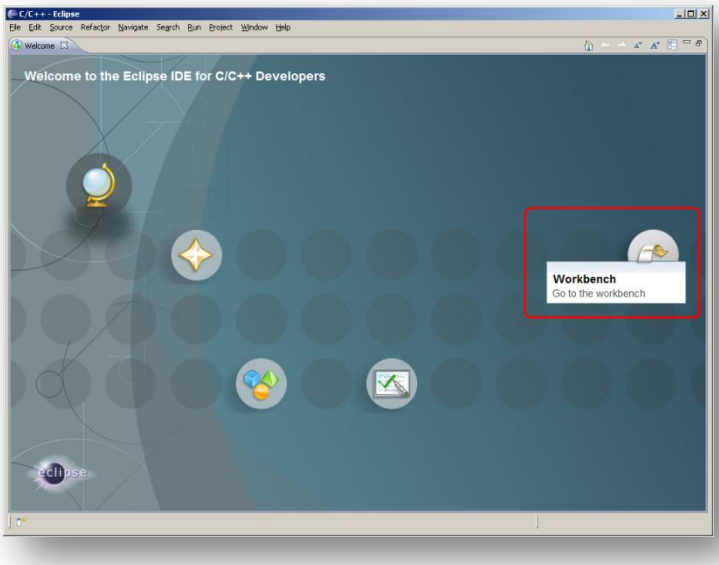
Una vez que se tienen todas las herramientas instaladas, llega el momento de su configuración en Eclipse. Para ello, abrimos Eclipse:



Tras la ventana de inicialización mostrada anteriormente, se presenta otra pantalla en la que hemos de indicar la carpeta que va a contener los proyectos que vamos a utilizar con Eclipse. Si vamos a utilizar la misma carpeta siempre, podemos marcar la opción que se muestra en la zona inferior con la cual el workspace se quedará ya definido y no se nos preguntará más en el arranque de Eclipse. Por el contrario si cada vez vamos a trabajar en carpetas distintas, no marcamos la opción inferior y en cada arranque de Eclipse se nos preguntará sobre el workspace. Una vez seleccionado el Workspace pulsamos sobre OK para que arranque Eclipse.

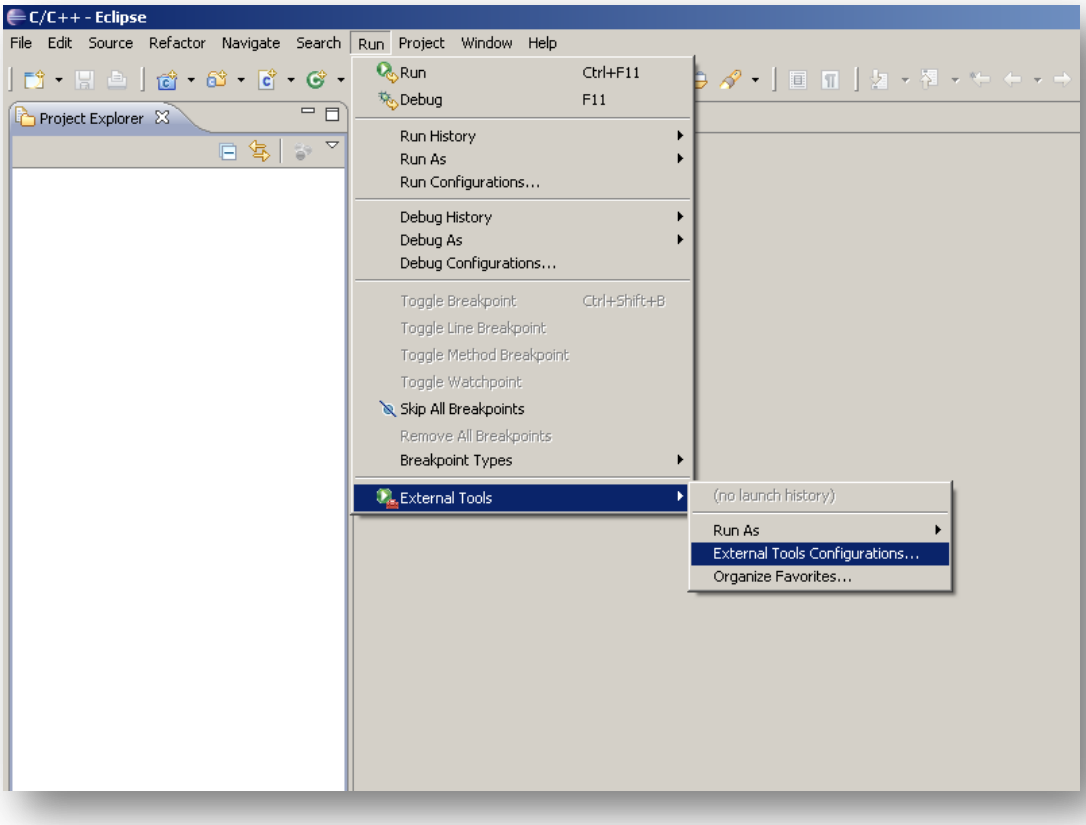


Esta es la página de bienvenida de Eclipse. Solo aparecerá la primera vez que se ejecute. Para continuar pulsar sobre la zona seleccionada.

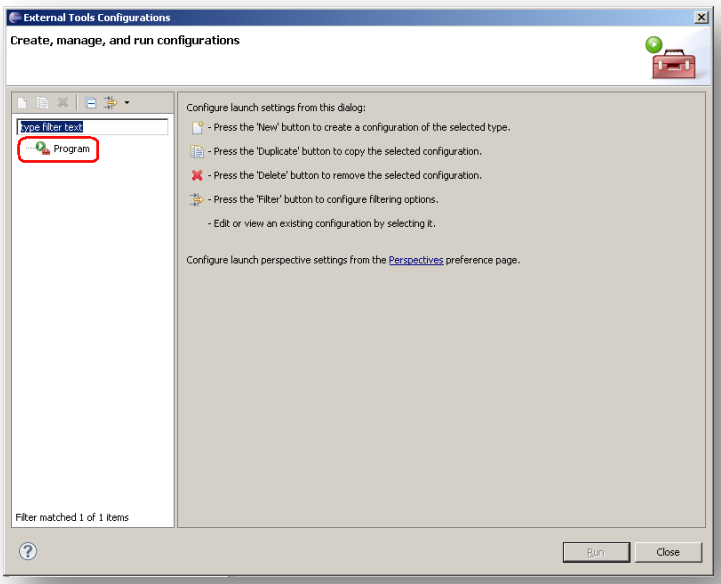


Lo primero que hay que hacer es configurar las herramientas externas que se van a utilizar desde Eclipse para el manejo del microprocesador.

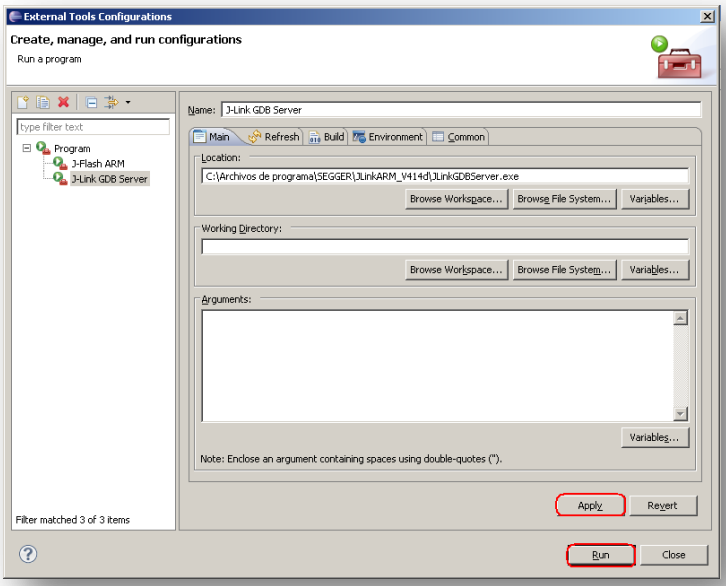
Pulsar sobre "**Run -> External Tools -> External Tools Configurations…**"



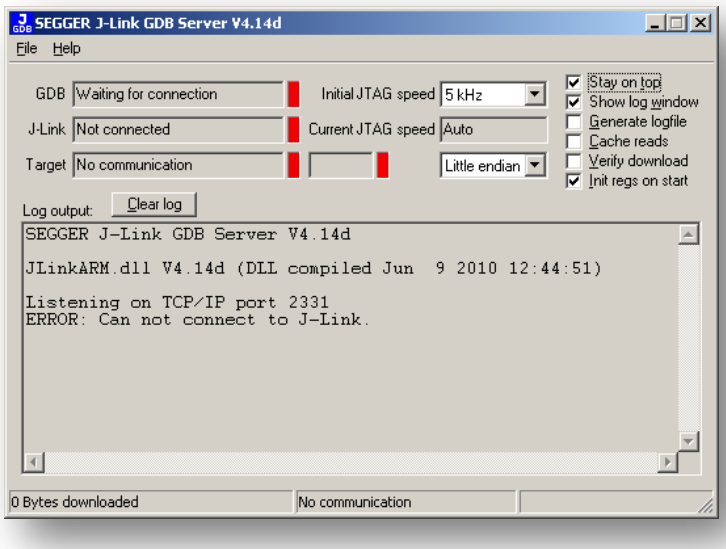
Y aparece la siguiente pantalla. En ella pulsar dos veces sobre **"Program"** para configurar la herramienta externa que se quiere utilizar.



Ahora pulsar una vez sobre **"New_Configuration"** y se van rellenando las distintas opciones. En "***Name"*** poner "J-Link GDB Server" (el nombre de la herramienta) y en "***Location"*** mediante **"Browse File System"** buscar donde se encuentra la aplicación. Si no se ha cambiado la carpeta de instalación por defecto de J-Link, el GDB Server se encuentra en "C:\Archivos de programa\SEGGER\JLinkARM_V414d\ JLinkGDBServer.exe". Una vez rellenos estos dos campos, se pulsa sobre **"Apply"** y luego sobre **"Close"**.



Si la configuración es correcta, al pulsar ***"Run"*** se debería abrir la siguiente aplicación:

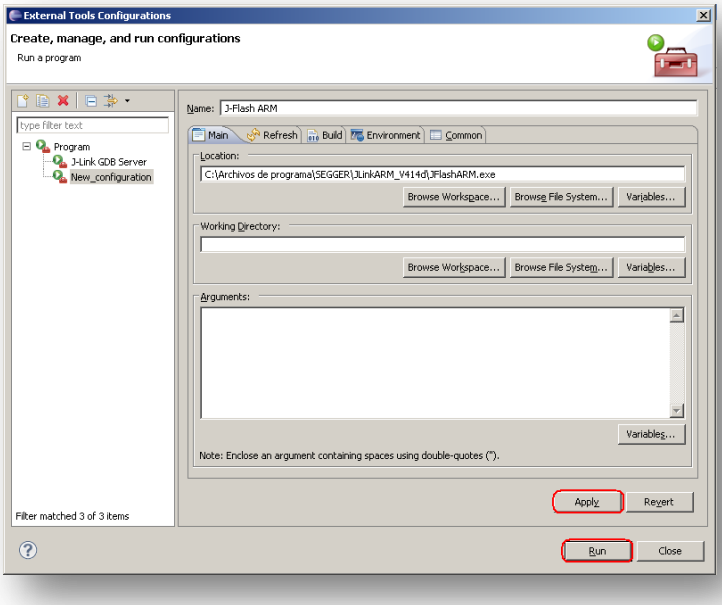


Ahora, se procede de la misma manera para la herramienta J-Flash ARM, pero hay que modificar **"Name"** y **"Location"**:

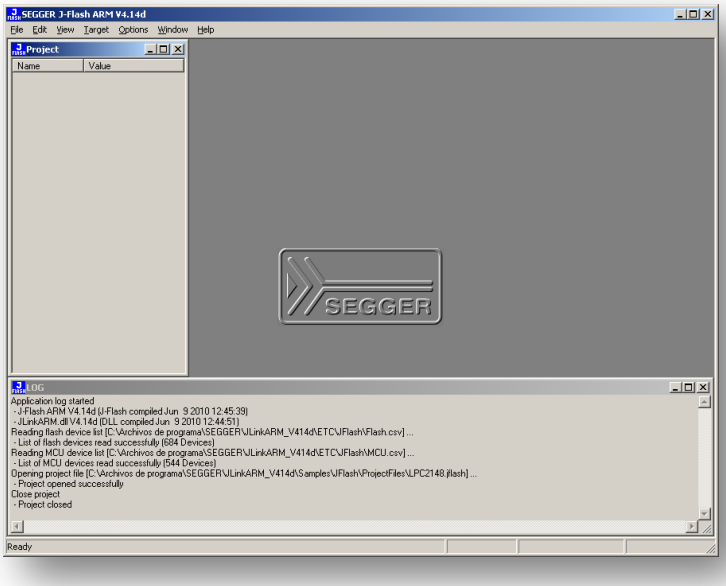
"Name": "J-Flash ARM"

"**Location":** dirigirse a la carpeta en la que se ha instalado los componentes de SEGGER y buscar la aplicación JFlashARM.exe. Si no se ha cambiado la carpeta de instalación por defecto de J-Link, la aplicación J-Flash ARM se encuentra en "C:\Archivos de programa\SEGGER\JLinkARM_V414d\JFlashARM.exe"

Después pulsar sobre "***Apply"*** y luego sobre "***Run"***:

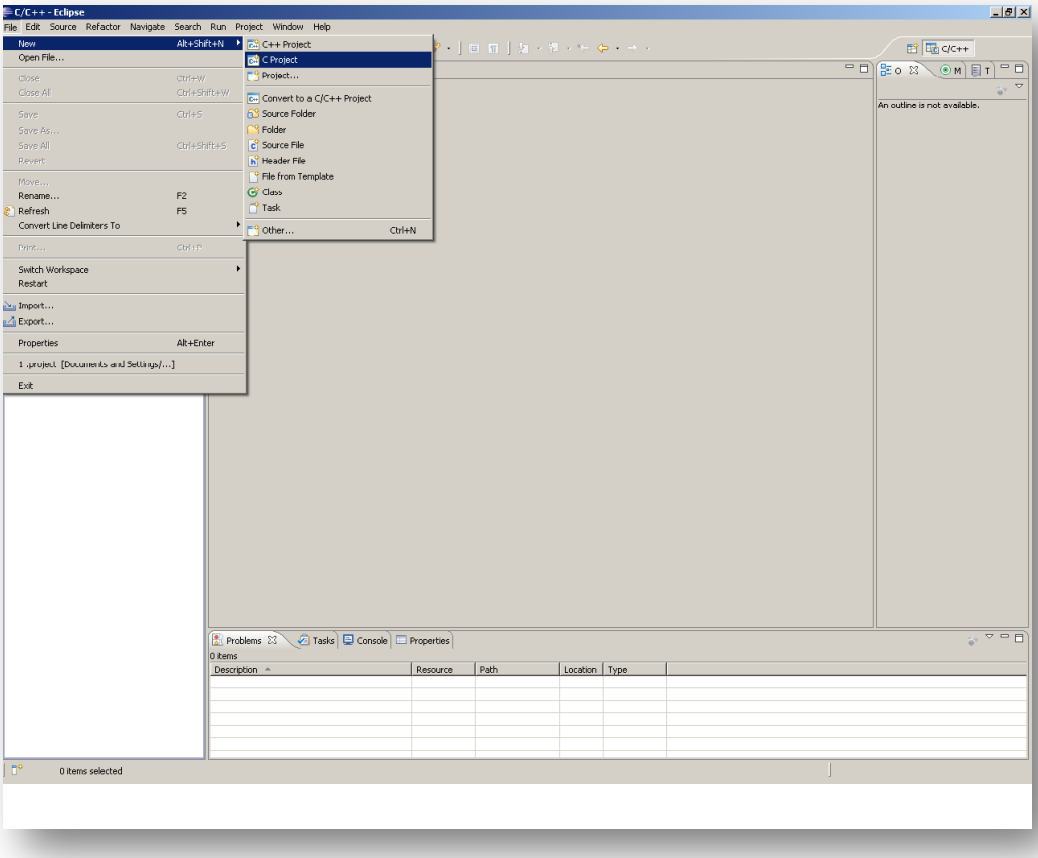


Si la ruta es correcta, al pulsar ***Run*** se debería abrir la siguiente aplicación:

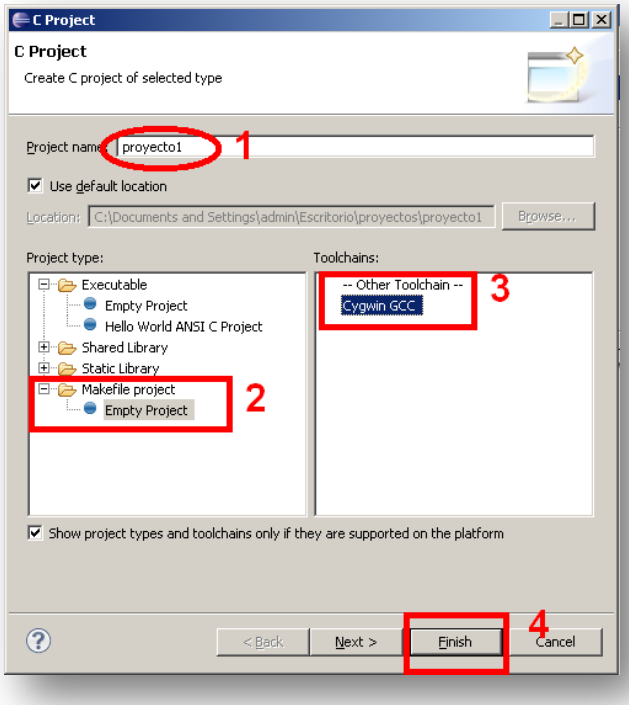


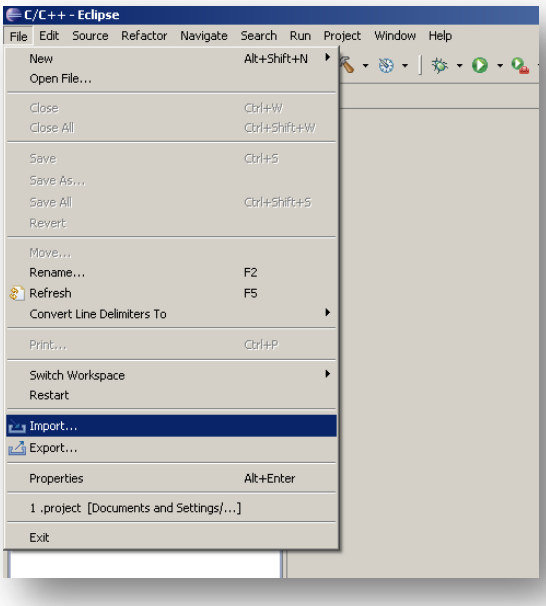
6. Creación de proyecto con Eclipse:

Para crear un nuevo proyecto con Eclipse pulsar sobre "***File -> New -> C Project".***

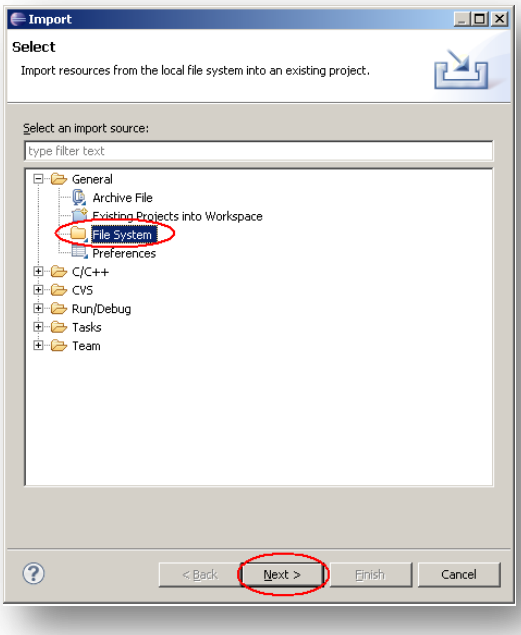


Aparecerá esta ventana donde hay que seleccionar en primer lugar el nombre del nuevo proyecto a crear. En segundo lugar desplegar el menú de Makefile Project y pulsar sobre **"Empty Project"**. En tercer lugar seleccionamos el tipo de toolchain, en nuestro caso, seleccionamos **"Cygwin gcc"**. Y por último pulsar sobre "***Finish"***.

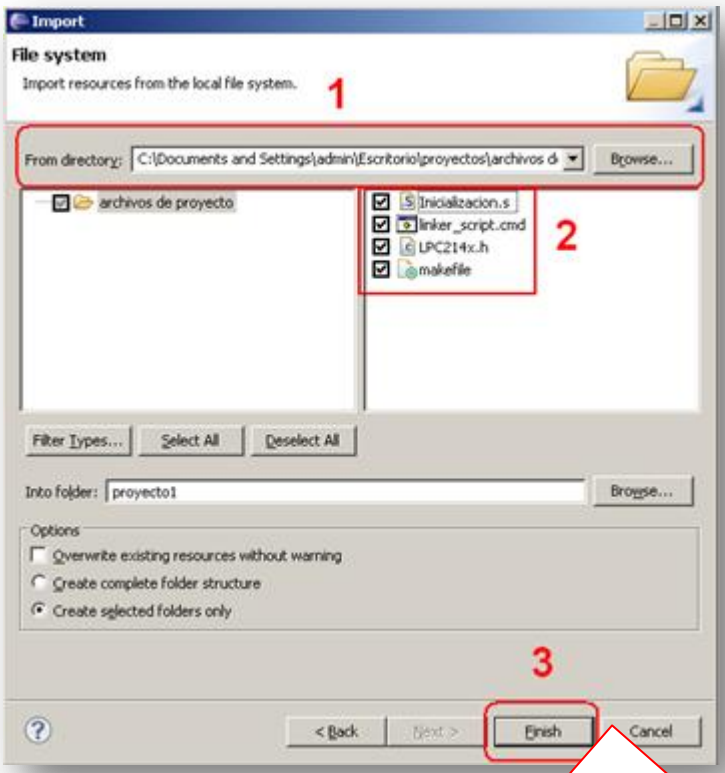




Ya tenemos nuestro proyecto creado, ahora hay que introducir los archivos necesarios, que son los que se adjuntan con el proyecto. Pulsamos sobre "***File -> Import***".



Ahora en esta nueva ventana pulsar sobre "***File System***" y luego sobre "***Next***".



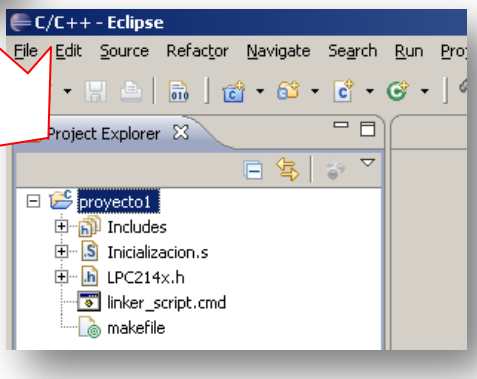
Ahora en esta nueva ventana seleccionar en primer lugar la ruta donde se encuentran los archivos necesarios.

En segundo lugar hay que asegurarse de que se importa al proyecto estos cuatro archivos:

-"Inicializacion.s"
-"linker_script.cmd"
-"LPC214x.h"
- "makefile"

Y por último pulsar sobre "***Finish"***.

Una vez que se ha pulsado sobre "***Finish"***, se puede comprobar (en la parte izquierda de la pantalla) como los archivos han sido añadidos a nuestro proyecto.



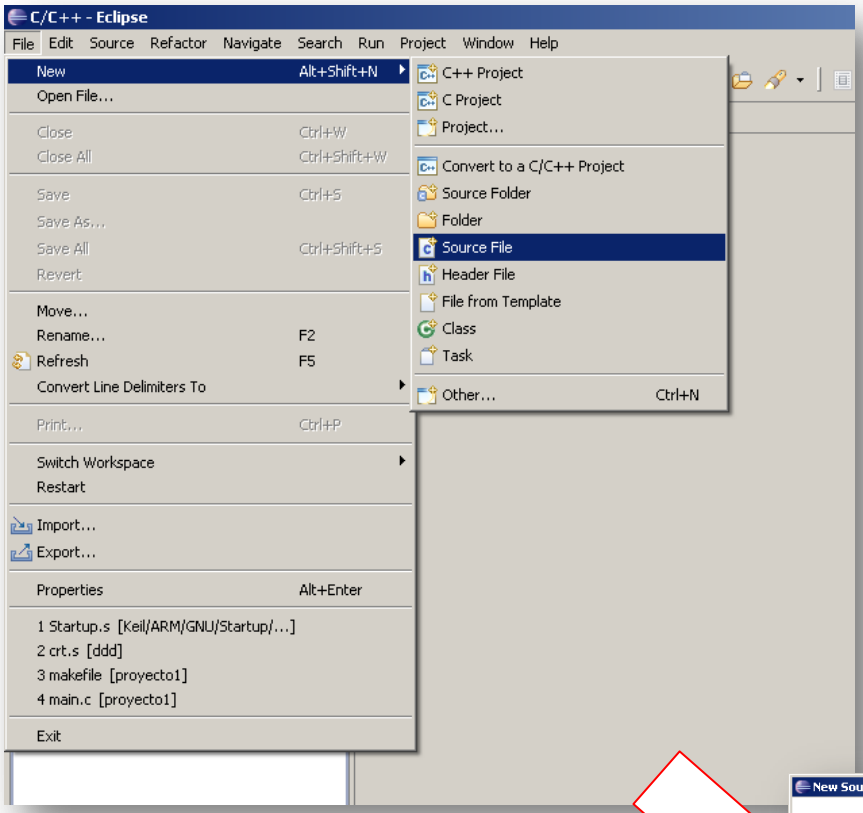
Hasta aquí, se han añadido los archivos necesarios para poder utilizar nuestro proyecto con herramientas GNU y en el microprocesador LPC2148. Ahora hay que añadir lo más importante que es el main del programa que vamos a ejecutar. Esto se puede hacer de dos formas distintas, o bien se importa de un main.c ya existente o bien crear uno nuevo.

Si nos encontramos en el primer caso, y ya tenemos el main.c creado en otra carpeta y queremos incorporarlo a nuestro proyecto, hay que hacer lo mismo que lo que se ha realizado para incorporar "Inicializacion.s", "linker_script.cmd", "LPC214x.h" y "makefile" pero como directorio fuente seleccionar la dirección en la que se

encuentre el main.c que queremos utilizar. Esta forma de incorporar archivos a nuestro proyecto es la que se debe emplear para añadirlas en el proyecto, y no pegarlos directamente en la carpeta en la que se encuentre el proyecto.

En el segundo de los casos, si lo que tenemos es que crear un nuevo main.c, el proceso a seguir es el siguiente:

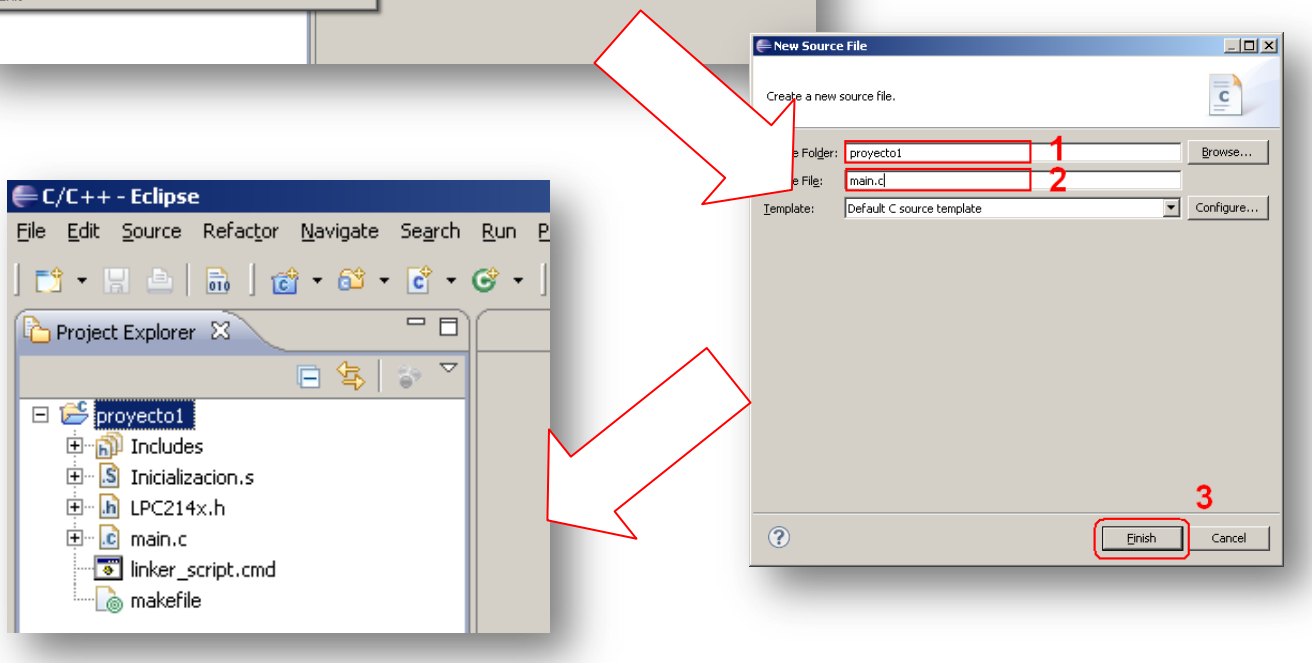
Pulsar sobre "***File -> New -> Source File".*** Y en la nueva ventana que aparece hay que rellenar los pasos unos y dos.



En el paso uno hay que escribir (Si no viene por defecto) el nombre del proyecto en el cual se va a crear el nuevo archivo.

En el paso dos hay que escribir el nombre del nuevo archivo con su extensión, en este caso, "***main.c.***

Por último se pulsa sobre ***"Finish"***.



En la ventana principal en el panel de la izquierda podemos observar si se ha añadido el archivo a nuestro proyecto correctamente.

En este punto, en el cual se tiene lo básico para que funcione nuestro proyecto, es un buen momento para pasar a explicar el contenido y la función de los ficheros que hemos importado al proyecto.

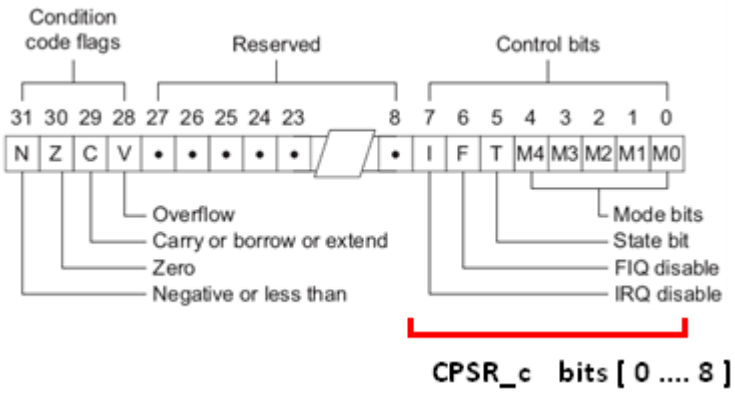
6.1 Descripción del archivo Inicializacion.s [14][15][16] [17]

Este archivo, se encarga de ejecutar las distintas instrucciones que son necesarias para inicializar el procesador y saltar al main.c.

En primer lugar se va a definir cuales son los distintos modos de funcionamiento del microprocesador. Este microprocesador, puede trabajar en siete modos distintos: User(modo de ejecución normal), FIQ(tratamiento de interrupciones FIQ) , IRQ(tratamiento de interrupciones IRQ), Undefined(se ha producido un fallo en la ejecución de una instrucción), Supervisor(Modo de protección para el sistema operativo), Abort(se ha producido un fallo en la carga de un dato o instrucción) y System(Modo de usuario privilegiado).

Para poder configurar los distintos modos, hay que modificar el registro CPSR (Current program status register) [17], pero no en su totalidad. Solo hay que manipular los ocho primeros bits para cambiar de modo de funcionamiento y activar, o desactivar las interrupciones IRQ y FIQ. Para solo manipular los ocho bits que a nosotros nos interesa se utiliza CPSR_c (Current program status register control bits).

En la siguiente figura puede verse la estructura del registro CPSR.



Bit 7: Puesto a 1, deshabilita las interrupciones IRQ
Bit 6: Puesto a 1, deshabilita las interrupciones FIQ
Bit 5: Puesto a 1, ejecuta en modo Thumb (16-bits) y puesto a 0 ejecuta en modo Arm (32-bits)
Del bit 4 al bit 0: modifica el modo de funcionamiento del procesador.

M[4:0]	Mode	Visible Thumb-state registers	Visible ARM-state registers
10000	User	r0–r7, SP, LR, PC, CPSR	r0–r14, PC, CPSR
10001	FIQ	r0–r7, SP_fiq, LR_fiq, PC, CPSR, SPSR_fiq	r0–r7, r8_fiq–r14_fiq, PC, CPSR, SPSR_fiq
10010	IRQ	r0–r7, SP_irq, LR_irq, PC, CPSR, SPSR_irq	r0–r12, r13_irq, r14_irq, PC, CPSR, SPSR_irq
10011	Supervisor	r0–r7, SP_svc, LR_svc, PC, CPSR, SPSR_svc	r0–r12, r13_svc, r14_svc, PC, CPSR, SPSR_svc
10111	Abort	r0–r7, SP_abt, LR_abt, PC, CPSR, SPSR_abt	r0–r12, r13_abt, r14_abt, PC, CPSR, SPSR_abt
11011	Undefined	r0–r7, SP_und, LR_und, PC, CPSR, SPSR_und	r0–r12, r13_und, r14_und, PC, CPSR, SPSR_und
11111	System	r0–r7, SP, LR, PC, CPSR	r0–r14, PC, CPSR

Nota: En el caso de utilizar una combinación de bits que no esté en la tabla, el procesador entrará en un estado impredecible y habrá que aplicar un reset, para volver a recuperarlo.

En estas primeras líneas de código, lo que se está realizando es asignar el tamaño de la pila para cada modo de funcionamiento del microprocesador. El tamaño signado a cada una de ellas es de 4 bytes.

```
/* Stack Sizes */
.set  UND_STACK_SIZE, 0x00000004
.set  ABT_STACK_SIZE, 0x00000004
.set  FIQ_STACK_SIZE, 0x00000004
.set  IRQ_STACK_SIZE, 0X00000004
.set  SVC_STACK_SIZE, 0x00000004
```

En el siguiente bloque de código se definen las distintas variables que contienen los bits necesarios para modificar el modo de funcionamiento del microprocesador.

```
.set  MODE_USR, 0x10       /* Modo de usuario normal*/
.set  MODE_FIQ, 0x11       /* Modo FIQ*/
.set  MODE_IRQ, 0x12       /* Modo IRQ*/
.set  MODE_SVC, 0x13       /* Modo supervisor*/
.set  MODE_ABT, 0x17       /* Modo abort*/
```

```
.set  MODE_UND, 0x1B       /* Modo de instrucción no definida*/
.set  MODE_SYS, 0x1F       /* Modo privilegiado*/
```

Si continuamos nos encontramos con la configuración de “I_bit” y “F_bit”, y los configuramos de forma que las interrupciones queden deshabilitadas.

```
.set  I_BIT, 0x80 /* cuando I_BIT está puesto a 1, las
interrupciones IRQ están dehabilitadas.
.set  F_BIT, 0x40 /* cuando F_BIT está puesto a 1, las
interrupciones FIQ están dehabilitadas.
```

Para poder comprender la siguiente parte del código, es necesario conocer algunas características de la arquitectura ARM.

```
.global     Reset_Handler
.global _startup
.func   _startup

_startup:


_vectors:

            ldr     PC, Reset_Addr
            ldr     PC, Undef_Addr
            ldr     PC, SWI_Addr
            ldr     PC, PAbt_Addr
            ldr     PC, DAbt_Addr
            nop   /* Reservado
            ldr     PC, [PC,#-0xFF0]
            ldr     PC, FIQ_Addr
```

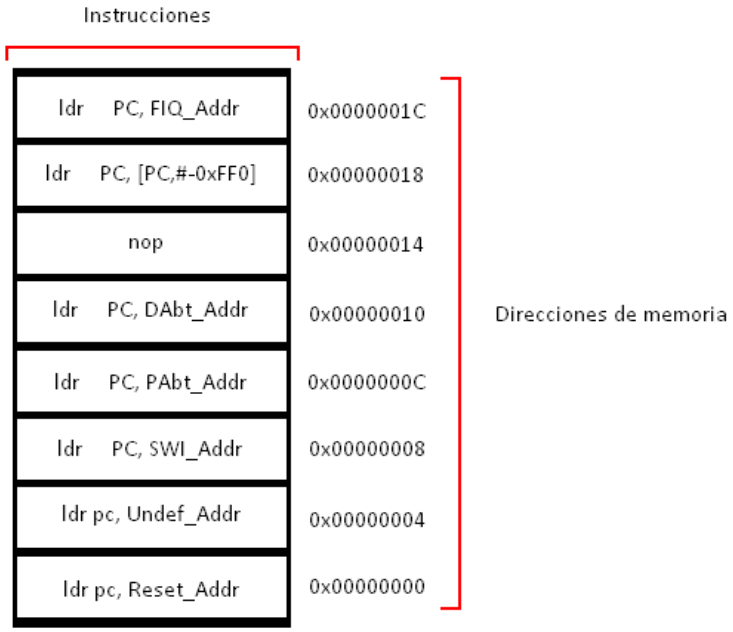
ARM exception vectors: [16]

Cuando una excepción es generada en el procesador, este deja lo que estaba haciendo y salta a una dirección de memoria fija que corresponde con el tipo de excepción que ha saltado. Estas direcciones de memoria fijas se llaman exception vectors y están mapeadas por defecto al inicio de la memoria (dirección 0x00000000). En la siguiente figura se muestra cuales son los exception vectors y cuáles son sus direcciones.

Table 3: ARM exception vector locations

Address	Exception
0x0000 0000	Reset
0x0000 0004	Undefined Instruction
0x0000 0008	Software Interrupt
0x0000 000C	Prefetch Abort (instruction fetch memory fault)
0x0000 0010	Data Abort (data access memory fault)
0x0000 0014	Reserved **Note:** Identified as reserved in ARM documentation, this location is used by the Boot Loader as the Valid User Program key. This is described in detail in "Flash Memory System and Programming" chapter on page 291.
0x0000 0018	IRQ
0x0000 001C	FIQ

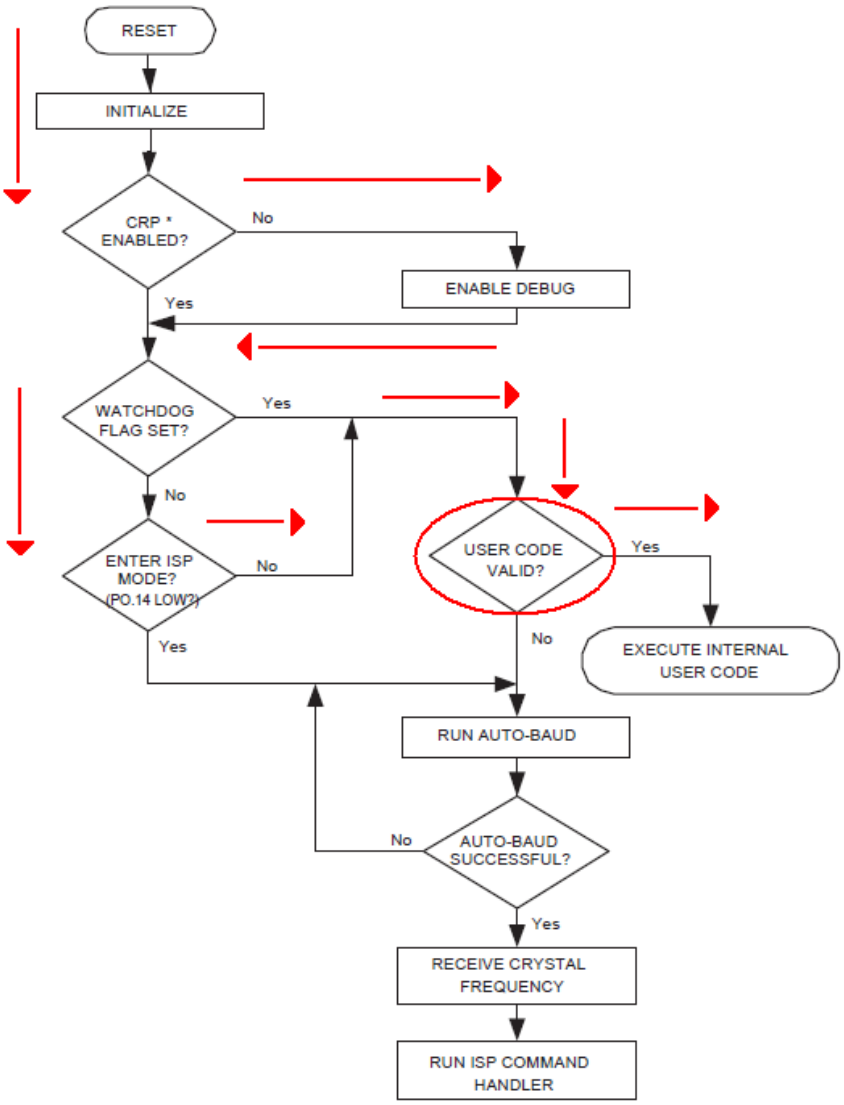
Como consecuencia de lo anteriormente mencionado la etiqueta Reset_Handler es el comienzo de nuestro código. Esto es debido a que nada mas arrancar el procesador, va a saltar la excepción de reset, por lo que el PC (program conunter) va a apuntar en la dirección 0x00000000, y en esta dirección de memoria se va a ejecutar la instrucción "`ldr PC, Reset_Addr`". Esta instrucción lo que hace es cargar el valor que contiene la etiqueta Reset_Addr en el PC y lo que contiene la etiqueta Reset_addr es la dirección de memoria de la etiqueta Reset_Handler.



De forma que gracias a esta sección de código, en cuanto se produzca una excepción, dependiendo del tipo, vamos a poder saltar a cualquier parte del código.

Valid User Program Key: [16]

El exception vector de la posición de memoria 0x00000014 debe contener el complemento a 2 del checksum del resto de los exception vectors. Esto va a hacer que el checksum total de los exception vectors sea 0, que es lo que se necesita para que el procesador deje ejecutar el código que se ha cargado en él.



* Code read protection

Luego el motivo por el cual se hace un NOP en la dirección 0x00000014, es porque la aplicación que se utilizamos para cargar los programas en el procesador (J-Flash ARM) ya se encarga de calcular e introducir el checksum por nosotros.

Instrucción: ldr PC, [PC,#-0xFF0]:[18] pág 70

Esta instrucción se encuentra en la posición de memoria 0x00000018, que es la que se encarga de las excepciones de tipo IRQ. Pues bien, cuando ejecutemos dicha instrucción, el valor de PC (Program Counter) será 0x18 + 0x8 = 0x20(ya que en modo ARM el valor de PC es el valor de la instrucción que estamos ejecutando mas 8).Luego al ejecutar esta instrucción lo que estamos realizando es restar a PC (0x20) el valor 0xFF0 y almacenarlo en el PC. Luego en PC estamos cargando la dirección 0x000000018 – 0xFF0 = 0x00000018 + 0XFFFFF010 = 0xFFFFF030. ¿Y qué tiene de especial esta dirección?. Pues el motivo por el cual se carga esa dirección es porque en ella es donde se encuentra el registro VicVectAddr, que contiene la dirección de rutina de servicio de la IRQ generada (como en este proyecto las interrupciones no están habilitadas, esta instrucción serviría como un primer paso a tratar las interrupciones IRQ)

Ahora pasamos a la última parte del código, esta parte es lo primero que va a ejecutar el microprocesador, como se ha mencionado anteriormente. Lo que se pretende con esta sección de código es configurar cada modo de funcionamiento y el tamaño de su pila(La etiqueta _stack_end, al igual que _etext, _data y _edata están definidos en el archivo linker_script, donde se explica cual es su utilidad.).

```
# Reset Handler

Reset_Handler:

                ldr   r0, =_stack_end

                msr   CPSR_c, #MODE_UND|I_BIT|F_BIT
                mov   sp, r0
                sub   r0, r0, #UND_STACK_SIZE
                msr   CPSR_c, #MODE_ABT|I_BIT|F_BIT
                mov   sp, r0
                sub   r0, r0, #ABT_STACK_SIZE
                msr   CPSR_c, #MODE_FIQ|I_BIT|F_BIT
                mov   sp, r0
                      sub   r0, r0, #FIQ_STACK_SIZE
                msr   CPSR_c, #MODE_IRQ|I_BIT|F_BIT
                mov   sp, r0
```

```
sub   r0, r0, #IRQ_STACK_SIZE
msr   CPSR_c, #MODE_SVC|I_BIT|F_BIT
mov   sp, r0
sub   r0, r0, #SVC_STACK_SIZE
msr   CPSR_c, #MODE_USR|I_BIT|F_BIT
mov   sp, r0
```

Observando el código se puede llegar a pensar que el registro sp (stack ponter) se está sobrescribiendo todo el tiempo. Pero resulta que no, pues todos los modos de funcionamiento menos el modo usuario tienen los registros sp (stack pointer) y lr (link register) privados. De forma que cada modo de funcionamiento va a tener su propio sp y lr.[17]

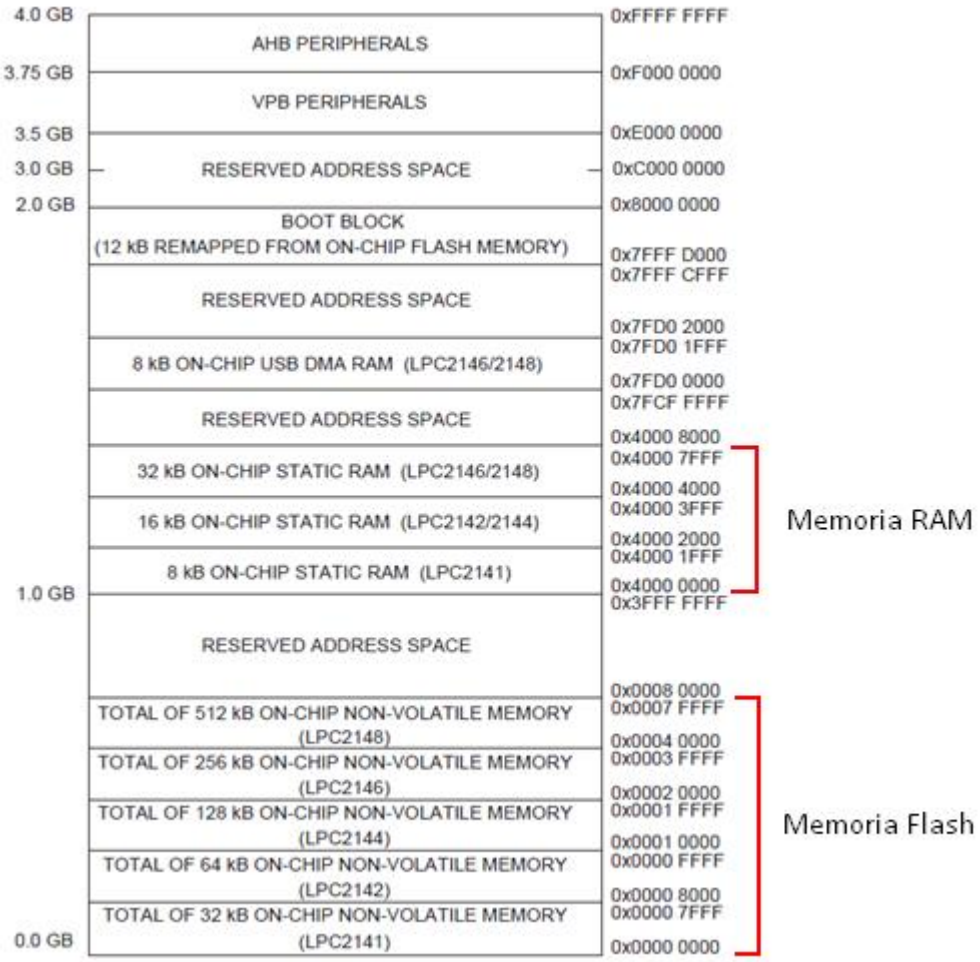
Y finalmente una vez que hemos configurado la pila para cado modo de funcionamiento, hay que copiar el contenido de .data alojado en la memoria flash, en la RAM e inicializar a cero la sección .bss antes de que se ejecute el código usuario.

```
                /* copiar la sección .data de la Flash a la RAM */
                ldr     R1, =_etext
                ldr     R2, =_data
                ldr     R3, =_edata
LoopRelocate:   cmp     R2, R3
                ldrlo   R0, [R1], #4
                strlo   R0, [R2], #4
                blo     LoopRelocate


                /* Inicializar a cero la sección .bss  */
                mov     R0, #0
                ldr     R1, =_bss_start
                ldr     R2, =_bss_end
LoopClear:      cmp     R1, R2
                strlo   R0, [R1], #4
                blo     LoopClear


                /* Saltar al main  */
                 b       main

.endfunc
.end
```

6.2 Descripción del archivo linker_script.cmd: [7]

El archivo linker_script se encarga de distribuir el código en la memoria del microprocesador de la forma que nosotros deseemos. Podemos cargar los programas en la memoria RAM, con la ventaja de que el programa irá más rápido, pero a cambio la información se perderá al quitar la alimentación del procesador. O bien podemos cargar el programa en la memoria flash, obteniendo una velocidad menor, pero a cambio sólo es necesario programarlo una vez pues la memoria flash no es volátil.

A continuación se muestra el mapa de memoria del LPC2148, va a ser la referencia a seguir a la hora de distribuir el código en el espacio de memoria.

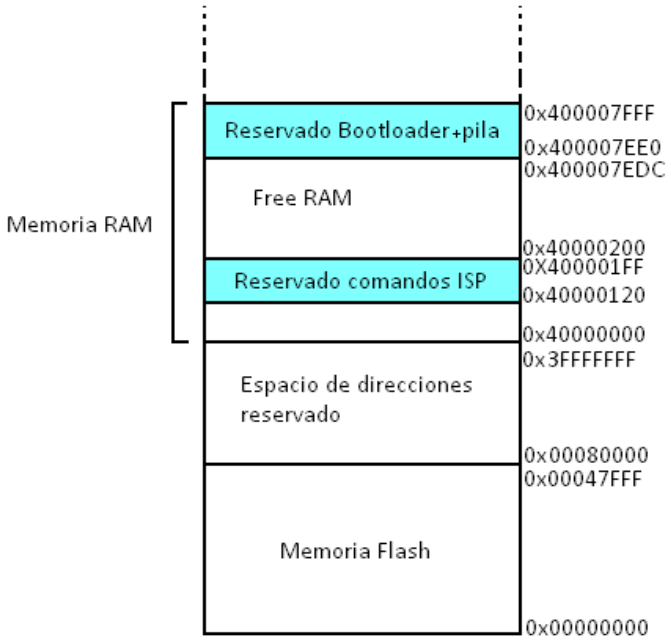


Antes de comenzar a detallar el código, hay que tener en cuenta que cualquier símbolo creado en el linker_script, es global, por lo tanto es accesible para el resto de programas del proyecto.

En el linker_script, se va a trabajar con segmentos de código, estas son generadas al compilar cada archivo. De forma que "Inicializar.s" va a tener sus propios segmentos de código, "main.c" también va a tener sus propios segmentos de código… etc. En este proyecto, se va a trabajar con los siguientes segmentos: ".text", ".data", ".bss", ".rodata".

- El segmento .text almacena el código ejecutable generado por el compilador.
- El segmento .data almacena los datos inicializados generados por el compilador.
- El segmento .bss almacena los datos no inicializados generados por el ensamblador.
- El segmento .rodata y .rodata* variables de solo lectura.
- El segmento glue_7 y glue_7t no se emplean en este proyecto, pero son necesarias a la hora de compilar.

Una vez que conocemos los segmentos de código, solamente queda colocarlos en las posiciones de memoria que sea necesario. Pero antes de ello, hay que tener en cuenta ciertas partes de la memoria que no pueden ser escritas, ya que su uso esta reservado a otros fines. Si se mira en [16] página 294, en los apartados 21.4.11 y 21.4.12 se indica que de la dirección 0x40000120 a la dirección 0x400001FF, su uso está reservado a los comandos ISP. Además los comandos de programación de la memoria flash están alojados en los últimos 32 bytes de la memoria RAM y justo debajo se reservan 256 bytes para el uso de la pila de comandos IAP y ISP, de forma que el esquema de memoria queda de la siguiente manera:



Por lo tanto, el área de libre programación de memoria flash queda comprendida entre las direcciones 0x00000000 -> 0x00047FFF y el área de libre programación de la memoria RAM queda comprendida entre 0x40000200 -> 0x400007EDC.

Ahora que se conocen las características descritas anteriormente, se procede a explicar el contenido del código:

En primer lugar, se define _startup como punto de entrada y después se definen los distintos bloques de memoria con su dirección de origen y su tamaño en bytes. También se define el símbolo global stack_end que indica la última dirección de memoria RAM programable.

```
ENTRY(_startup)


MEMORY
{
flash                       : ORIGIN = 0,          LENGTH = 512K
reservado_comando_ISP       : ORIGIN = 0x40000120, LENGTH = 223
ram                         : ORIGIN = 0x40000200, LENGTH = 32513
reservado_botloader_pila    : ORIGIN = 0x40007FE0, LENGTH = 32
}

_stack_end = 0x40007EDC;
```

Mediante el comando **"SECTIONS"**, se le indica al linker como han de fusionarse las secciones y en que lugar de memoria van a ser alojadas. Tras ello, hay que colocar el location counter(.) a las dirección de memoria 0x0. Esto es necesario pues el location counter es el que se encarga de almacenar la dirección de memoria en la cual hemos de programar.

Una vez que hemos configurado el location counter, se procede a crear las secciones de memoria de salida, estas son .codigo_ejecutable, .datos_inicializados y .datos_no_inicializados.

La primera sección es .codigo_ejecutable y contiene las secciones generados por el compilador .text .rodata y glue_7. Esta sección se encuentra alojada en la memoria flash.

```
SECTIONS
{
	. = 0;

	.codigo_ejecutable :
	{
		*Inicializacion.o (.text)
		*(.text)
		*(.rodata)
		*(.rodata*)
		*(.glue_7)
		*(.glue_7t)
		_etext = .;
	} >flash
```

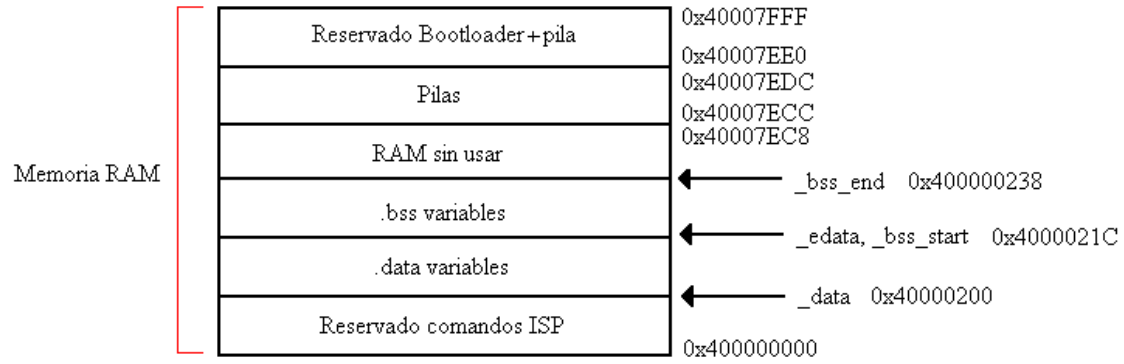
La siguiente sección, .datos_inicializados, contiene como su nombre indica todas las variables que han sido inicializadas en cualquier fichero del proyecto y va alojada en la memoria RAM y en la memoria flash. En la memoria flash se carga la LMA (Load Memory Addres) de la sección .data, es decir donde reside la información. Y en la memoria RAM lo que se introduce es la VMA (Virtual Memory Address) que es la dirección en la que se cargará la sección .data al arrancar el programa, en el startup. El motivo por el cual se carga en la memoria flash y en la RAM es porque si el procesador pierde la alimentación, la sección .datos inicializados se perdería y el programa solo con la sección de código no funcionaría.

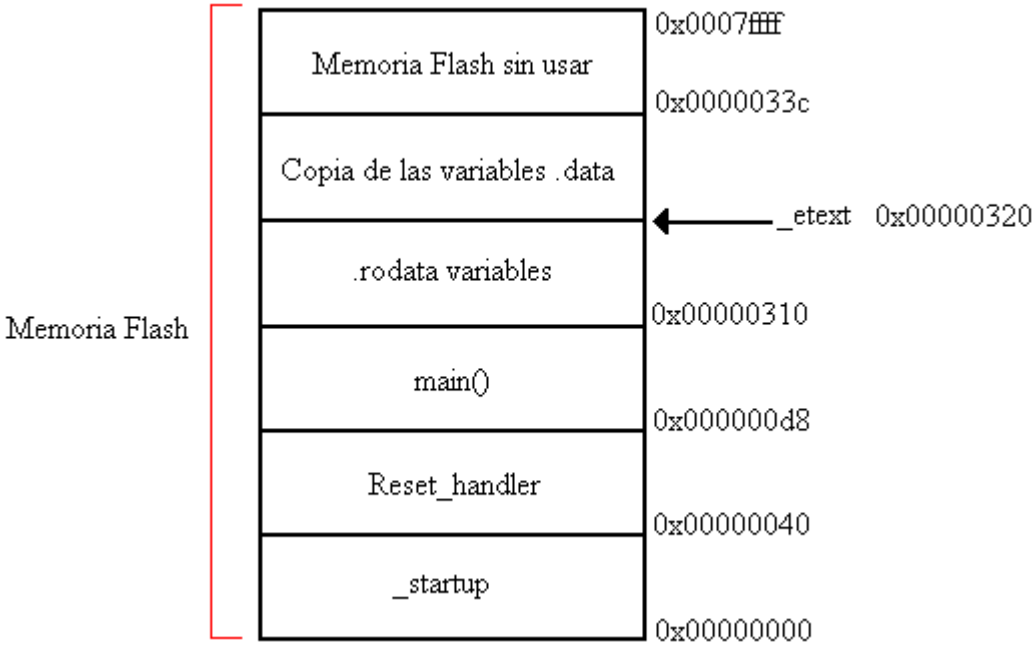
```
	.datos_inicializados :
	{
		_data = .;
		*(.data)
		_edata = .;
	} >ram AT >flash
```

Por último, se crea la sección .datos_no_inicializados que también va cargada en la memoria RAM y se crean las variables globales _bss_end que almacena el final de la sección .datos_no_inicializados y la variable _end que indica la última dirección de memoria programada.

```
.datos_no_inicializados :
{
      _bss_start = .;
      *(.bss)
} >ram

. = ALIGN(4);
_bss_end = . ;
}
_end = .;
```

De forma que el mapa de memoria queda dibujado de la siguiente manera:





6.3 Descripción del archivo makefile [9]

El archivo makefile es llamado por la herramienta Make que se encuentra alojada dentro de Cygwin. En él se va a detallar el orden en el que se llama al compilador, enlazador,… etc. además de sus opciones de configuración.

En primer lugar se crea la etiqueta "NAME" que contiene el nombre del proyecto al que pertenece este makefile, y "ARCHIVOS_C_DEL_PROYECTO" contiene todos los archivos *.c que queremos que formen parte del proyecto.

```
NAME   = proyecto1
ARCHIVOS_C_DEL_PROYECTO        = main.c funcion_encendido.c
```

En el segundo bloque de código, se encarga de crear etiquetas con las llamadas al compilador, enlazador, depurador, herramienta objcopy y herramienta objdump. Además se configuran los flags que acompañan a la llamada de cada herramienta. El motivo por el cual se crean etiquetas con el contenido de los flags de configuración es el de simplificar el código y hacerlo más legible.

```
compilador          = arm-elf-gcc
linking             = arm-elf-ld -v
ensamblador         = arm-elf-as
copiar              = arm-elf-objcopy
volcar              = arm-elf-objdump
```

Llamadas a las herramientas

```
compilador_flags    = -I./ -c -fno-common -O0 -g
ensamblador_flags   = -ahls -mapcs-32 -o Inicializacion.o
linking_flags       = -Map main.map -Tlinker_script.cmd -o main.out
copiar_flags        = -O ihex
volcar_flags        = -x --syms
```

Flags de configuración

Descripción de los flags de configuración del compilador [5]:

- **-I./:** especifica los directorios donde buscar primero los archivos a compilar. En este caso, como los archivos a compilar se encuentran en el mismo directorio que el proyecto, se selecciona el directorio raíz del proyecto.

- **-c**: opción para no invocar el enlazador, ya que esto se realiza más adelante.

- **-fno-common**: aloja a las variables no inicializadas en la sección .bss.

- **-O0**: establece el nivel de optimización del código. Para poder depurar es obligatorio tener esta opción en el valor 0.

- **-g**: produce información para el depurador.

Descripción de los flags de configuración del ensamblador: [19]

- **-ahls**: flag que se utiliza para generar la información que va a aparecer en Inicializar.lst. Gracias a él, se incluye información de código de alto nivel, incluye el ensamblaje y los símbolos.

- **-mapcs-32**: selecciona la llamada de 32 bits

- **-o Inicializar.o**: crea el archivo objeto "Inicializar.o" que contiene la salida del ensamblador.

Descripción de los flags de configuración del linker [6]:

- **-Map main.map**: se crea el archivo main.map que contiene la información sobre la dirección que ocupa cada parte del código en memoria y el tamaño que ocupa. También nos ayuda a saber si algunas variables van en la sección .data o .rodata, al igual que también indica si hay alguna sección con contenido que no se ha añadido en el "linker_script.cmd".

- **-Tlinker_script.cmd**: identifica el nombre del archivo que va a utilizar el linker, en este caso "linker_script.cmd".

- **-o main.out:** crea el archivo objeto main.out que contiene la salida del linker.

Descripción de los flags de configuración de objcopy [8]:

- **-o ihex**: indica el formato del fichero de salida, en este caso hexadecimal.

Descripción de los flags de configuración de objdump [20]:

- **-x**: a la hora de crear el archivo main.dmp incluye en él todos los archivos de cabecera incluyendo la tabla de símbolos y si hay alguna relocalización de código.
- **--syms**: muestra la tabla de símbolos con un determinado formato.

Una vez que se han definido las etiquetas con las llamadas a las herramientas GNU y los flags de configuración, es cuando el makefile comienza a llamar a las herramientas.

Con la primera línea, se establece la regla de inicio del makefile, es decir que cuando se invoque a la herramienta make y esta abra el fichero makefile, al ejecutar la sentencia all: test, va a realizar todas las operaciones de las que se compone el makefile, ya que para ejecutar la regla test, esta necesita el archivo main.out, y para obtener el archivo main.out es necesario tener Inicializacion.o, main.o y linker_script.cmd. El linker_script siempre va a estar disponible pues es uno de los archivos que se adjuntan con el proyecto, pero para obtener Inicializacion.o y main.o, es necesario ejecutar sus propias sentencias.

Sentencia de entrada: la sentencia entrada simplemente llama a la sentencia test.

```
all: test
```

Sentencia de test: en esta sentencia se crea los archivos de salida main.hex, que es el archivo que se programa en la memoria del microprocesador y main.dmp que es el archivo que contiene la información del fichero objeto main.out. Como parámetros de

entrada tiene main.out pero antes hay que generarlo (Se crea en la siguiente sentencia).

```
test: main.out
	@echo "……copiar"
	$(copiar) $(copiar_flags) main.out main.hex
	$(volcar) $(volcar_flags) main.out > main.dmp
```

Sentencia de main.out: esta sentencia se encarga de unir los archivos main.o y Inicializacion.o dando como resultado el archivo main.out que es el que se utiliza en la sentencia de test. Como parámetros de entrada tiene Inicializacion.o, main.o y linker_script.cmd. Inicializacion.o y main.o se crean en las sentencias posteriores.

```
main.out: Inicializacion.o main.o linker_script.cmd
	@echo "……unir"
	$(linking) $(linking_flags) -o main.out  Inicializacion.o main.o
```

Sentencia de Inicializacion.o: mediante esta sentencia se crea el archivo Inicializacion.o como resultado de ensamblar el archivo Inicializacion.s.

```
Inicializacion.o: Inicializacion.s
	@echo "……ensamblando"
	$(ensamblador) $(ensamblador_flags) Inicializacion.s
```

Sentencia de main.o: la siguiente sentencia se encarga de compilar el archivo main.c dando lugar al archivo main.o.

```
main.o: main.c
	@echo "……compilando"
	$(compilador) $(compilador_flags) $(ARCHIVOS_C_DEL_PROYECTO)
```

Sentencia de clean: cuando se invoca a la sentencia clean desde Eclipse, lo que se está realizando es llamar al comando –rm e introducirle como parámetros de entrada los ficheros que se deseen eliminar.

```
clean:
	-rm -f Inicializacion.lst funcion_encendido.o main.elf Inicializacion.o
	main.o main.out main.hex main.map main.dmp
```

6.4 Descripción del archivo lpc214x.h

En este archivo se asocian las direcciones de memoria de los registros del procesador a etiquetas con el nombre del registro, de forma que facilita el uso de los registros del procesador al no ser necesario acordarse de la dirección del registro. Simplemente hay que conoce su nombre.

7. Ejemplo práctico.

Para comprobar el correcto funcionamiento de la toolchain, se ha creado un main que se encarga de configurar la frecuencia cclk del procesador, tras ello se crean varias variables que son modificadas durante la ejecución del programa en la memoria flash y por último se llama al archivo función_encendido.c que se encarga de hacer parpadear un led.

7.1 Archivo main.c

En el archivo main.c es donde se va a escribir el programa que queremos ejecutar en el procesador. Este archivo está escrito en el lenguaje de programación C.

Para este proyecto, se ha realizado un main muy simple, pero que permite comprobar que el sistema cumple con su cometido. El programa que se ha desarrollado en el main se encarga de configurar el cclk, crear variables, modificarlas, configurar el P0.11 para poder activar y desactivar el led y por último llamar al archivo función_encendido.c.

La creación y modificación de variables no supone ninguna dificultad, simplemente se les asignan unos valores iniciales y según se va ejecutando el programa, dicho valores se van variando.

El siguiente paso consiste en llamar a al método initialize(). Este método se encarga de configurar la frecuencia cclk[16] del procesador.

Teniendo en cuenta que la placa del LPC2148 posee un reloj de 12.0 Mhz y que se desea que funcione a la velocidad de 60.0 Mhz (ha de ser múltiplo de 12Mhz), si nos dirigimos al manual del LPC2148 [16], página 28, se explica cómo obtener la frecuencia deseada mediante P y M.

Table 21: Elements determining PLL's frequency

Element	Description
F_{OSC}	the frequency from the crystal oscillator/external oscillator
F_{CCO}	the frequency of the PLL current controlled oscillator
CCLK	the PLL output frequency (also the processor clock frequency)
M	PLL Multiplier value from the MSEL bits in the PLLCFG register
P	PLL Divider value from the PSEL bits in the PLLCFG register

Si continuamos leyendo en la página anteriormente citada, se indica algunas restricciones sobre los valores de las frecuencias que pueden tomar Fosc, CCLK y Fcco:

- Fosc ha de permanecer dentro del rango comprendido entre 10 Mhz y 25 Mhz.
- CCLK ha de permanecer en el rango comprendido entre 10 Mhz y 60 Mhz.
- Fcco ha de permanecer dentro del rango comprendido entre 156 Mhz y 320 Mhz.

En este punto es conocido Fosc que es la frecuencia del oscilador, (en este caso es de 12.0 Mhz) y también se conoce el valor de CCLK (en este caso es 60.0 Mhz). Luego queda por hallar el valor de M, P. Para ello, se utilizan las siguientes fórmulas, obtenidas de la página 33 del lpc214x user manual [16]:

- CCLK = M x Fosc.
- CCLK = Fcco / (2 * P).

Para calcular el valor de M, se utiliza la primera fórmula, de forma que $M = \frac{CCLK}{Fosc}$, luego M= 60 Mhz / 12 Mhz = 5. Pero como en el registro PLLCFG hay que escribir M-1, se escribe un 4 en los bits asignados al multiplicador M, luego PLLCFG[4 : 0] = 00100.

Para calcular P hay que seguir un proceso similar pero con la segunda fórmula $P = \frac{Fcco}{CCLK \times 2}$ Sustituyendo los valores de Fcco mínimo y máximo, vamos a

obtener el valor de P que haga que Fcco esté dentro del rango.

(1) P = 156 Mhz / (60 Mhz x 2) = 1.3

(2) P = 320 Mhz / (60 Mhz x 2) = 2.6

Luego el valor entero de P que hace que “Fcco” permanezca dentro de su rango (156 Mhz <-> 320 Mhz) es 2. Lo mismo que ocurre con “M” ocurre con “P”, y es que en el registro “PLLCFG”, también hay que escribir P-1, luego escribiendo en los 2 bits asignados al divisor “P”, PLLCFG [5 : 6] = 01.

Finalmente el valor que ha de tomar el registro PLLCFG es 0b00100100 = 0x24. Pero para que adquiera dicho valor, hay que llamar a la función “feed()” que se encarga de escribir la secuencia “PLLFEED = 0xAA”; “PLLFEED = 0x55”; necesaria para poder modificar los registros “PLLCON” y “PLLCFG”. Una vez realizado esto, se habilita el PLL mediante el registro PLLCON

```
PLLCON=0x1;
feed();

// Esperar a que el PLL se estabilice
while(!(PLLSTAT & PLOCK)) ;

// conectar el PLL como la fuente de reloj
PLLCON=0x3;
feed();
```

Para terminar con la función “initialize()”, se activa la función MAM, que hace que el acceso a los datos de memoria sea óptimo, se configura el registro MAMTIM que se encarga de seleccionar el número de ciclos CCLK necesarios para acceder a la memoria flash (ya que evidentemente esta es mas lenta que el microcontrolador) y finalmente, se configura el reloj de los periféricos (pclk) para que sea el mismo que el de el procesador (CCLk).

```
MAMCR=0x2;
MAMTIM=0x4;
VPBDIV=0x1;
```

Cuando se termina de ejecutar el método Initialize(), hay que configurar el pin 11 del pinsel0 como GPIO , para ello, no hay que modificar nada pues al inicializar el procesador, se pone por defecto como GPIO. El siguiente paso consiste en configurar el pin 11 como pin de salida, ya que por defecto se inicializa como entrada. Esto se realiza con el comando IODIR0= |0x00080000. Una vez que ya tenemos configurado el pin, el siguiente paso consiste en llamar a funcion_encendido que es la que se encarga del manejo del led.

7.2 Descripción del archivo función_encendido.c

El archivo función_encendido.c de lo que se encarga es de encender el led, esperar 500ms y apagarlo durante un bucle infinito.

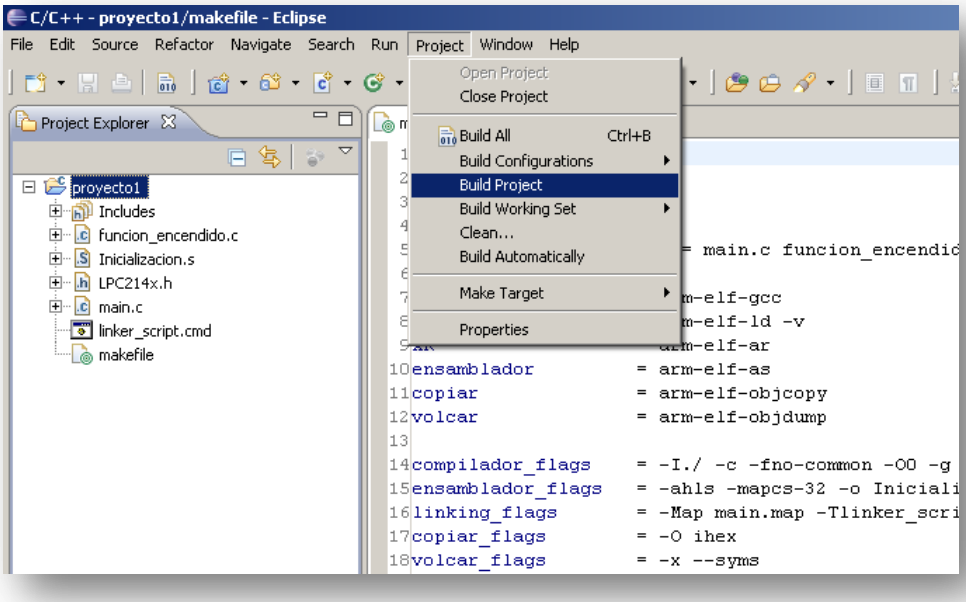
```
	void encender(){
	int j=0;
	while (1) {
		for (j = 0; j < 500000; j++ );		//tiempo de espera
		FIO0SET = 0x00000800;			//led on
		for (j = 0; j < 500000; j++ );		//tiempo de espera
		FIO0CLR = 0x00000800;			//led off
	}
}
```

7.3 Compilación y unión del proyecto:

Ahora que ya se conoce la utilidad de todos los archivos que se adjuntan en el proyecto, se procede a su compilación.

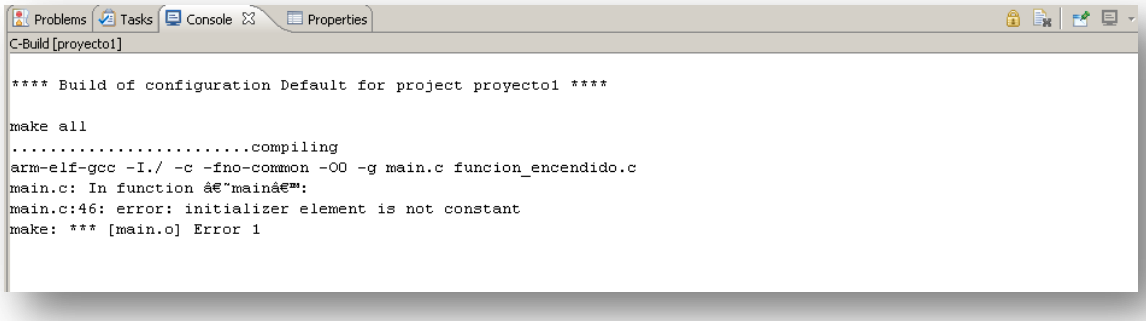
Partiendo de la base que se tiene creado el proyecto con todos los ficheros importados en la carpeta del proyecto, el proceso consiste en:

Para compilar el proyecto, se pulsa sobre **"Project -> Build Project"**



Y en la salida de consola, se nos indica las operaciones realizadas y en el caso de que se produzca algún error de compilación se muestra por consola y se detiene el proceso indicando en qué archivo está el error y cual es el motivo del fallo.

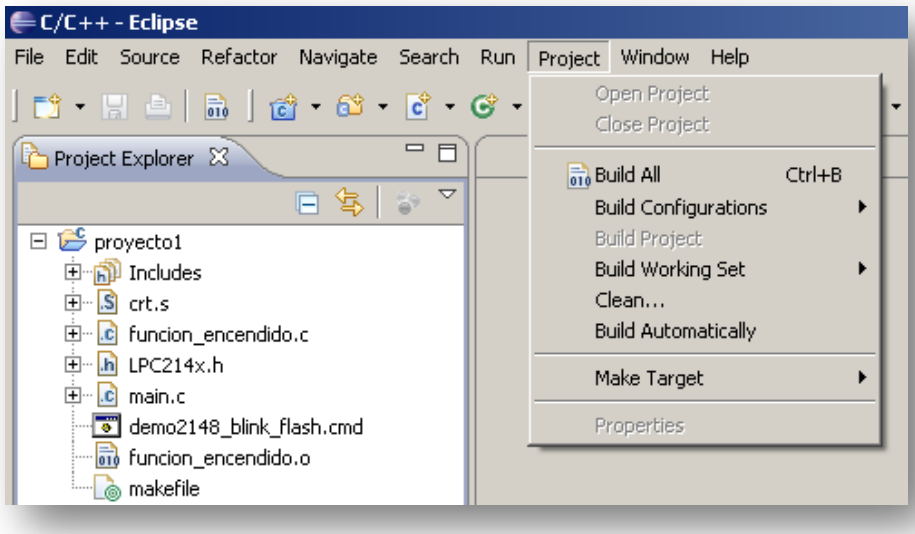
Este es un ejemplo de un fallo en el que nos indica que en la línea 46 del archivo main.c el elemento inicializado en dicha línea no se ha inicializado con una constante.



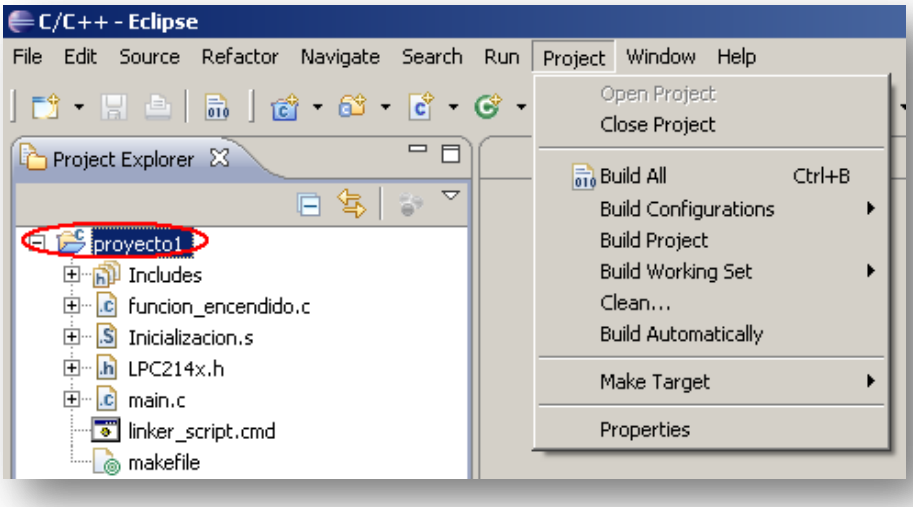
Una peculiaridad que hay que anotar es que si se recompila un proyecto compilado recientemente y no ha sido modificado ninguno de los ficheros fuente, entonces no se vuelve a ejecutar todo el makefile, simplemente se vuelve a repetir solo la sentencia principal (**"test"**), ya que los archivos necesarios para la ejecución de la sentencia test ya están creados y no han sido modificados.

Otro detalle a destacar es que a veces no está disponible la opción **"Build Project"**, y esto es debido a que no se tiene seleccionado el proyecto en su fichero raíz, es decir:

No se puede generar el proyecto.



Si se puede generar el proyecto.



7.4 Grabación del programa creado en la memoria flash

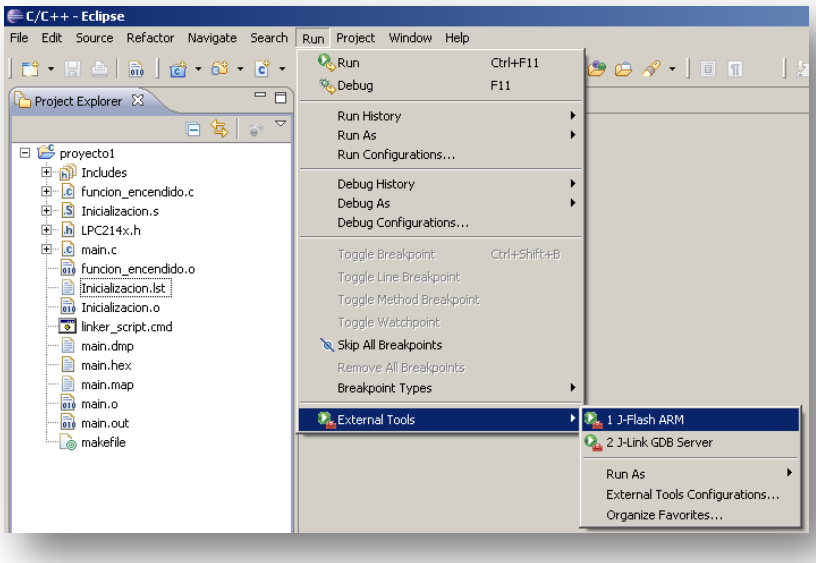
Una vez que se ha compilado y unido el proyecto, se obtienen una serie de nuevos archivos. Estos son:

- Archivos de información: Se encargan de mostrar cómo se ha distribuido el código por la memoria, en qué dirección se encuentra cada instrucción… etc. Estos archivos son “Inicializacion.lst”, “main.dmp” y “main.map”

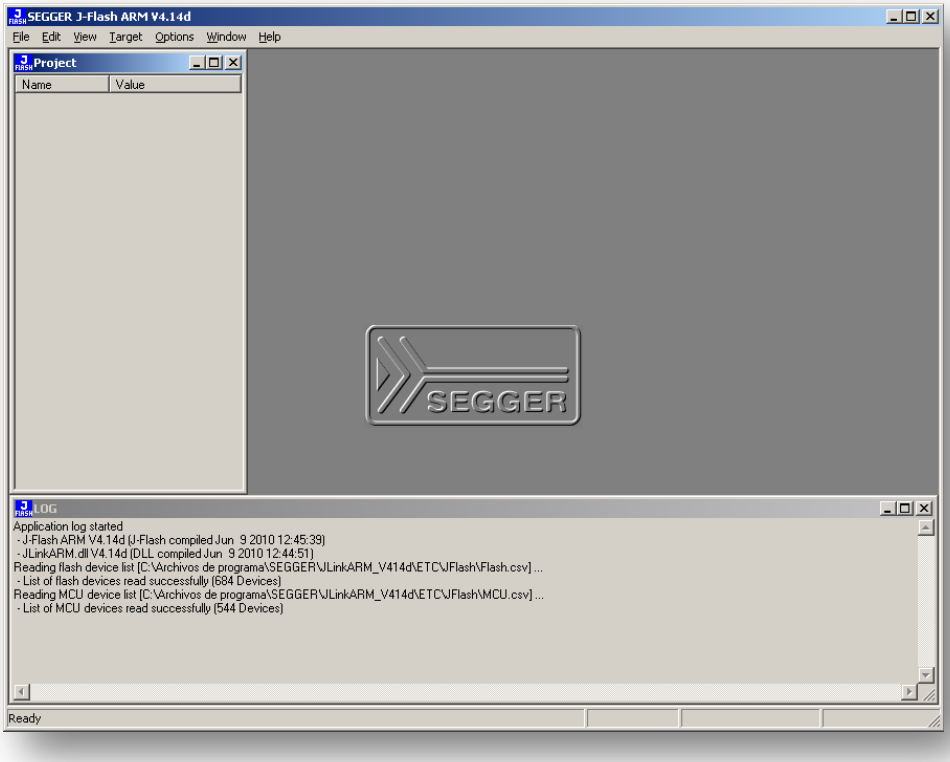
- Archivos objeto: Son archivos objeto, es decir, están preparados para ser interpretados por el microprocesador, y estos son: Inicializacion.o, main.o, main.out. y main.hex

El archivo objeto que nos interesa es main.hex, ya que va a ser el que se cargue en la memoria flash del microprocesador. ¿Cómo cargar dicho archivo en la memoria del microprocesador?. La respuesta es J-Flash ARM [10], herramienta que ya está instalada y que se puede ejecutar desde Eclipse. De forma que los pasos para cargar el programa en la memoria flash son los siguientes:

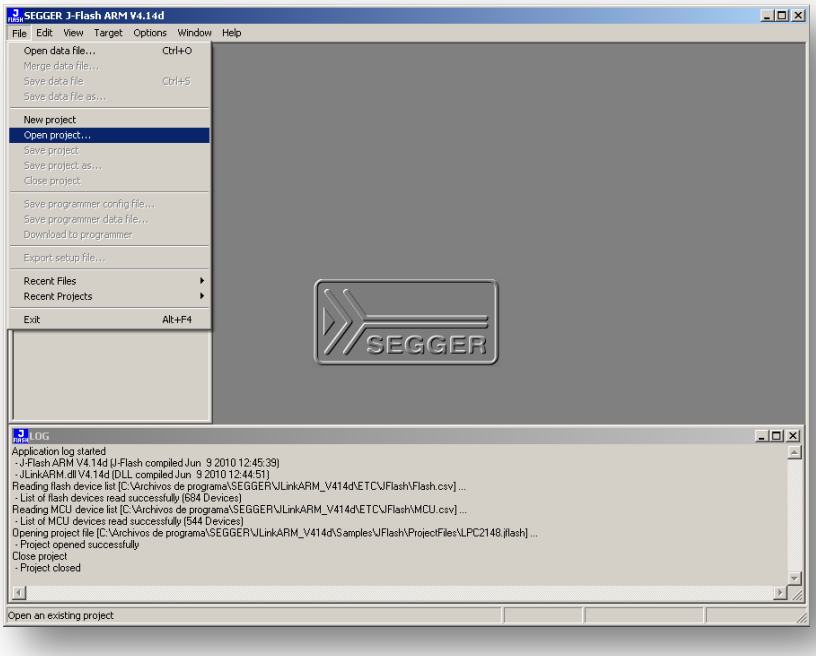
En primer lugar se abre la herramienta externa “J-Flash ARM” mediante Eclipse: **“Run -> External Tools-> J-Flash ARM”.**



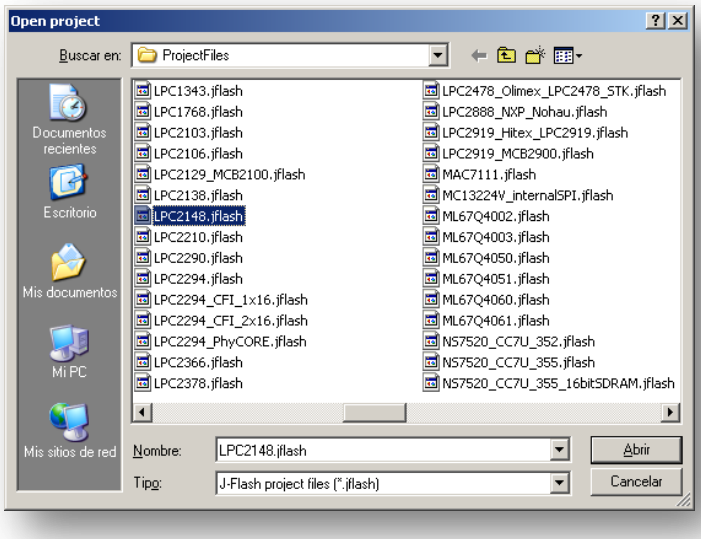
Es entonces cuando se abre la aplicación:



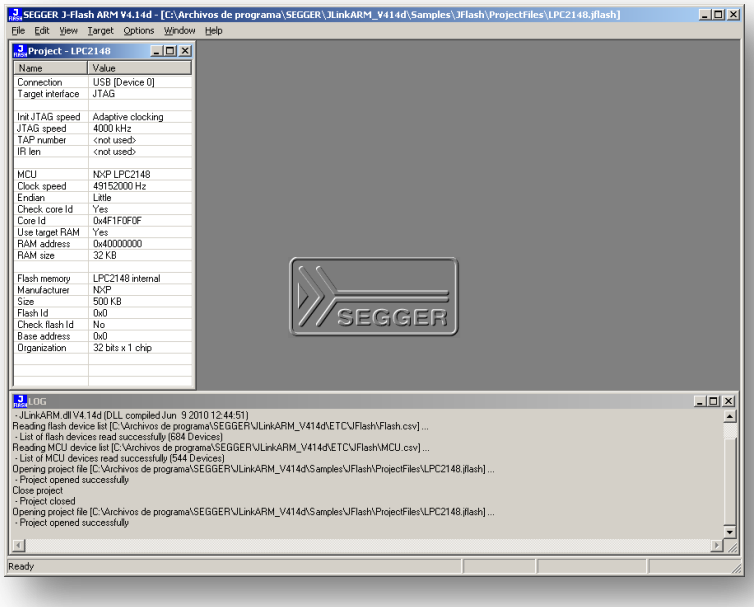
Ahora el siguiente paso consiste en configurar J-Flash ARM para poder comunicarse con el LPC2148, para ello, se pulsa sobre “File - > Open Project”.



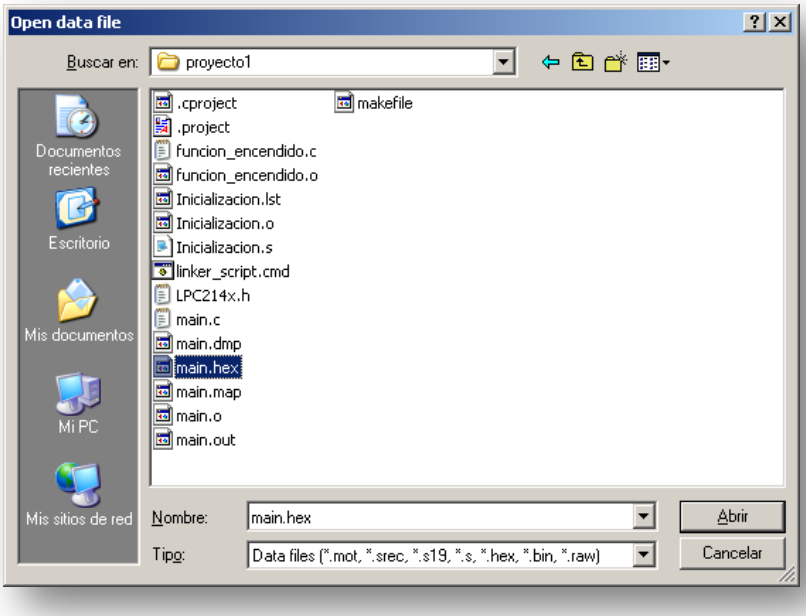
Y en la ventana que se acaba de abrir, seleccionar el microprocesador que se esté utilizando, en este caso “**LPC2148.jflash**” y se pulsa sobre **“Abrir”.**



Ahora en el lado izquierdo, aparecen algunas de las características de programación del microprocesador.

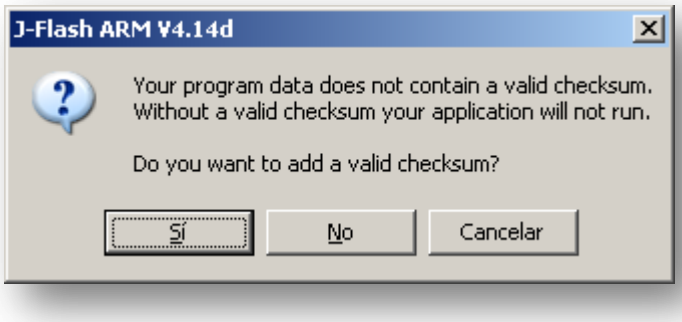


Pues bien, hasta este punto ya tenemos configurada la herramienta J-Flash ARM para poder programar al LPC2148. Lo próximo que hay que hacer es cargar el archivo main.hex, para ello se pulsa sobre **"File -> Open data file"** y nos dirigimos al fichero que contenga el archivo main.hex que queremos cargar.



Una vez cargado, para programarlo, hay que:

- Pulsar sobre **"Target -> Connect"** para establecer contacto con el procesador.
- Pulsar sobre **"Target -> Erase sectors"** para eliminar cualquier tipo de información que esté alojada en la memoria del procesador.
- Pulsar sobre **"Target -> Program & Verify"** para programar el contenido de main.hex en la memoria. Al hacer esto, nos va a preguntar que si se introduce un checksum válido pues no contiene ningún checksum válido y le tenemos que indicar que si. Esto es completamente normal que nos lo pida. Si nos dirigimos al apartado "7.1.1Descripción del archivo Inicializacion.s" en la página 61 de este manual, se habla sobre dicho checksum y ya se indicó que era esta aplicación la que lo introducía en la posición 0x00000014 de la memoria.



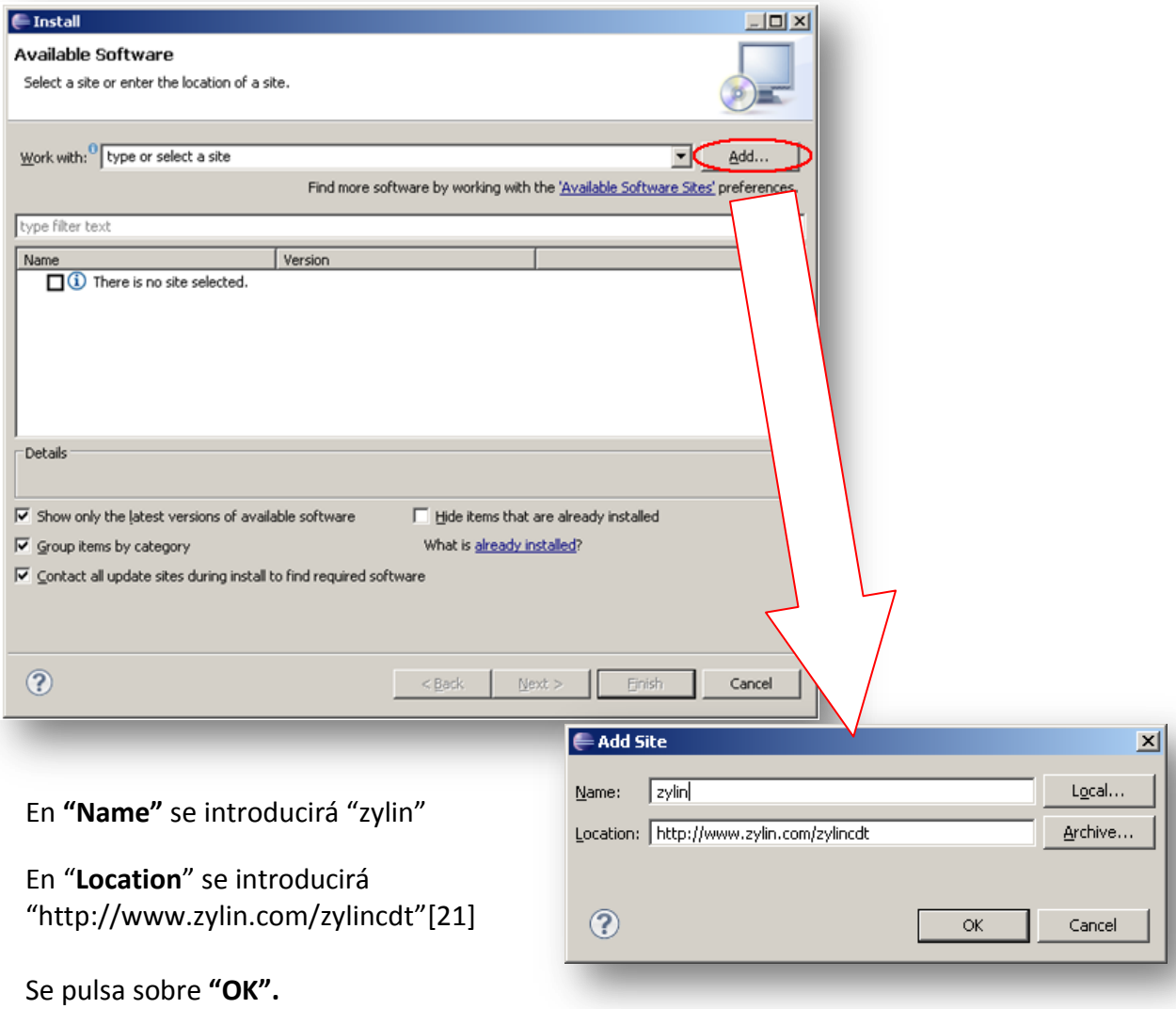
- Una vez cargado el programa nos informa de que se ha cargado correctamente y ahora, para que el programa comience a funcionar, hay que pulsar sobre **"Target -> Start Application"** y se puede comprobar cómo el led comienza a parpadear.

Al estar cargado el programa en la memoria flash, si se le desconecta la alimentación al LPC2148 y luego se le vuelve a conectar, el programa seguirá funcionando, como era de esperar.

7.5 Depuración del proyecto

En este apartado se va a mostrar los pasos necesarios para configurar Eclipse de forma que sea capaz de depurar el programa obtenido: **"main.out"**. Antes de nada hay que instalar el plugin Zylin CDT, ya que Eclipse CDT tiene dificultad para trabajar con los depuradores remotos y este plugin lo soluciona.

Para ello, se abre eclipse, se pulsa sobre **"Help -> Install New Software"** y se abre la siguiente ventana. Pulsaremos sobre **"Add"** aparecerá otra ventana donde hay que poner el nombre de la aplicación a instalar y la dirección de descarga. En este caso la única forma de obtener el plugin Zylin CDT es a través de Internet.

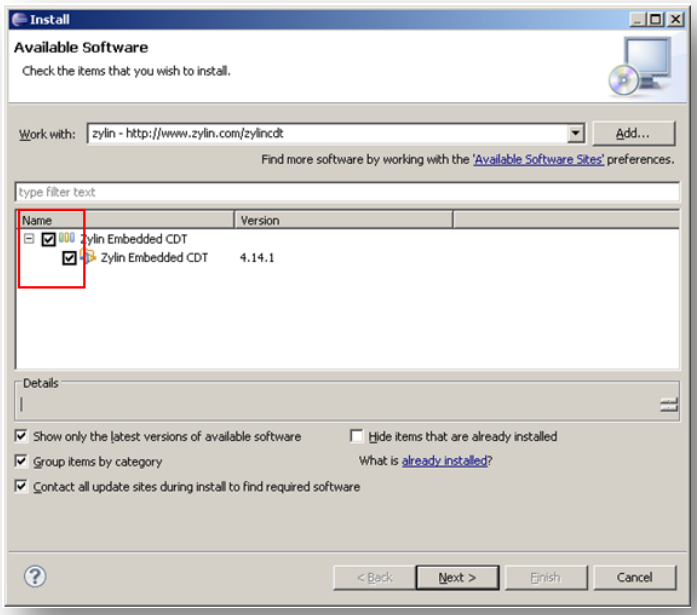


En **"Name"** se introducirá "zylin"

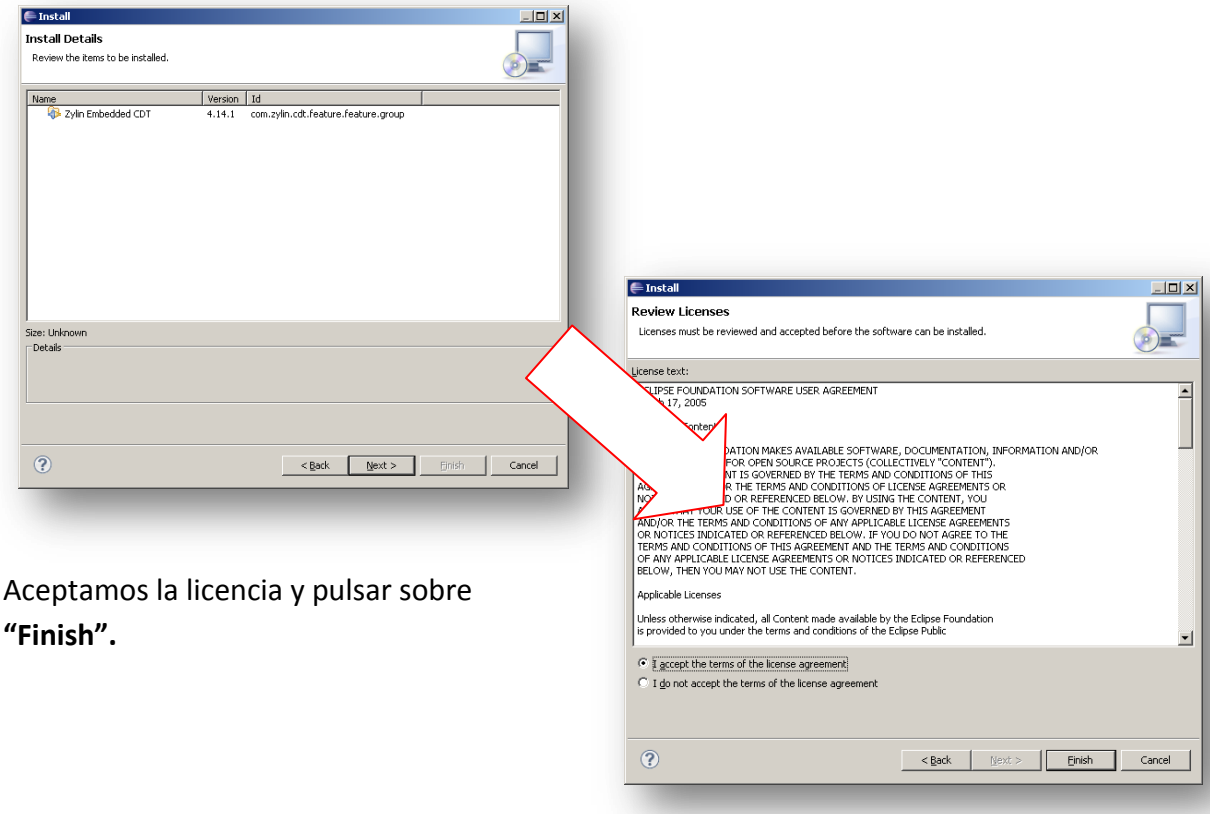
En "**Location**" se introducirá "http://www.zylin.com/zylincdt"[21]

Se pulsa sobre **"OK".**

Y ya nos aparece el plugin listo para ser instalado. Seleccionamos el plugin a instalar y se pulsa **“Next”.**



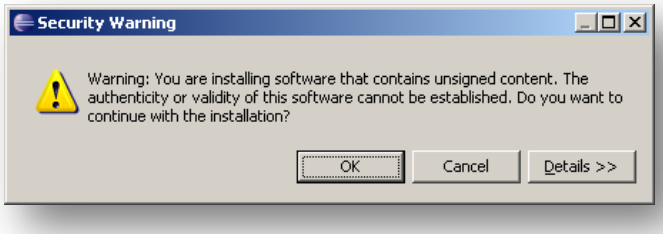
Ahora aparece una nueva ventana en la que se nos indica los detalles de la instalación. Pulsamos sobre **“Next”.**



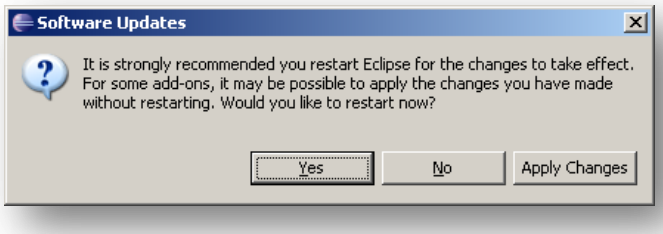
Aceptamos la licencia y pulsar sobre **“Finish”.**

Momento en el que empezará la instalación del plugin.

Durante la instalación va a aparecer una ventana indicando que se va a instalar un código que no ha sido firmado y que por lo tanto no se puede verificar su autenticidad, pulsamos sobre **"OK".**

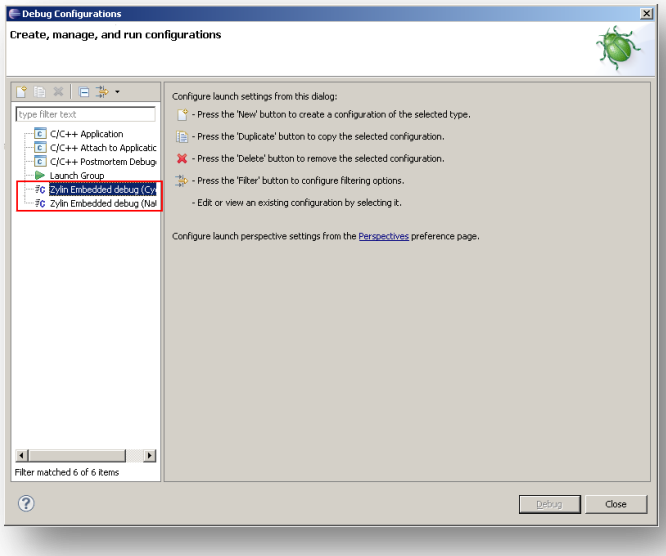


Una vez que ha terminado la instalación se nos ofrece la opción recomendada de reiniciar Eclipse para finalizar la instalación, pulsamos sobre **"Yes"** y la próxima vez que iniciemos Eclipse, ya tendremos instalado el plugin Zylin cdt**.**

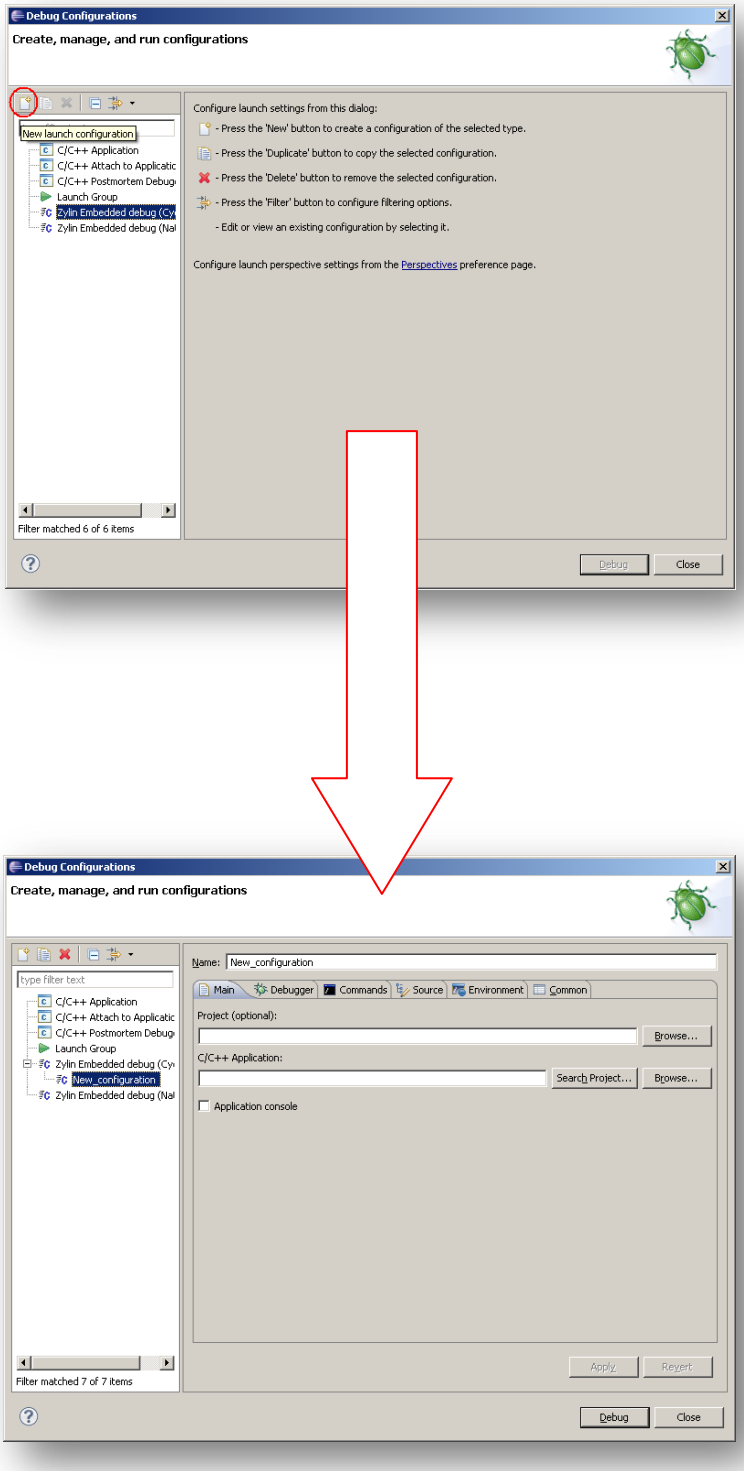


Para comprobar la correcta instalación, pulsar sobre la flecha de al lado de la opción debug  y en el menú desplegable que aparece, seleccionar **"Debug Configurations"**.

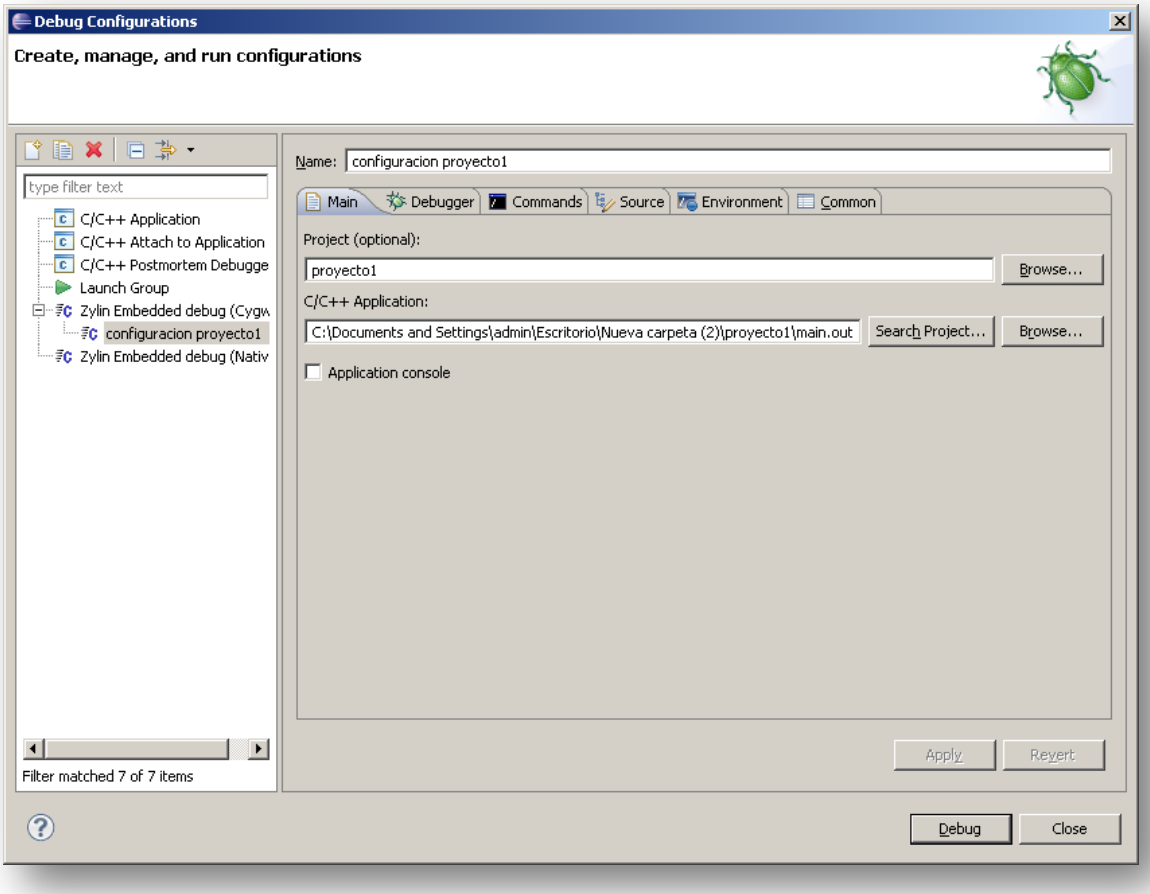
Si el plugin ha sido instalado correctamente, entonces nos debe aparecer las opciones **"Zylin Embedded debug (Cygwin)"** y **"Zylin Embedded debug (Native)"** que son las marcadas en rojo en la imagen inferior.



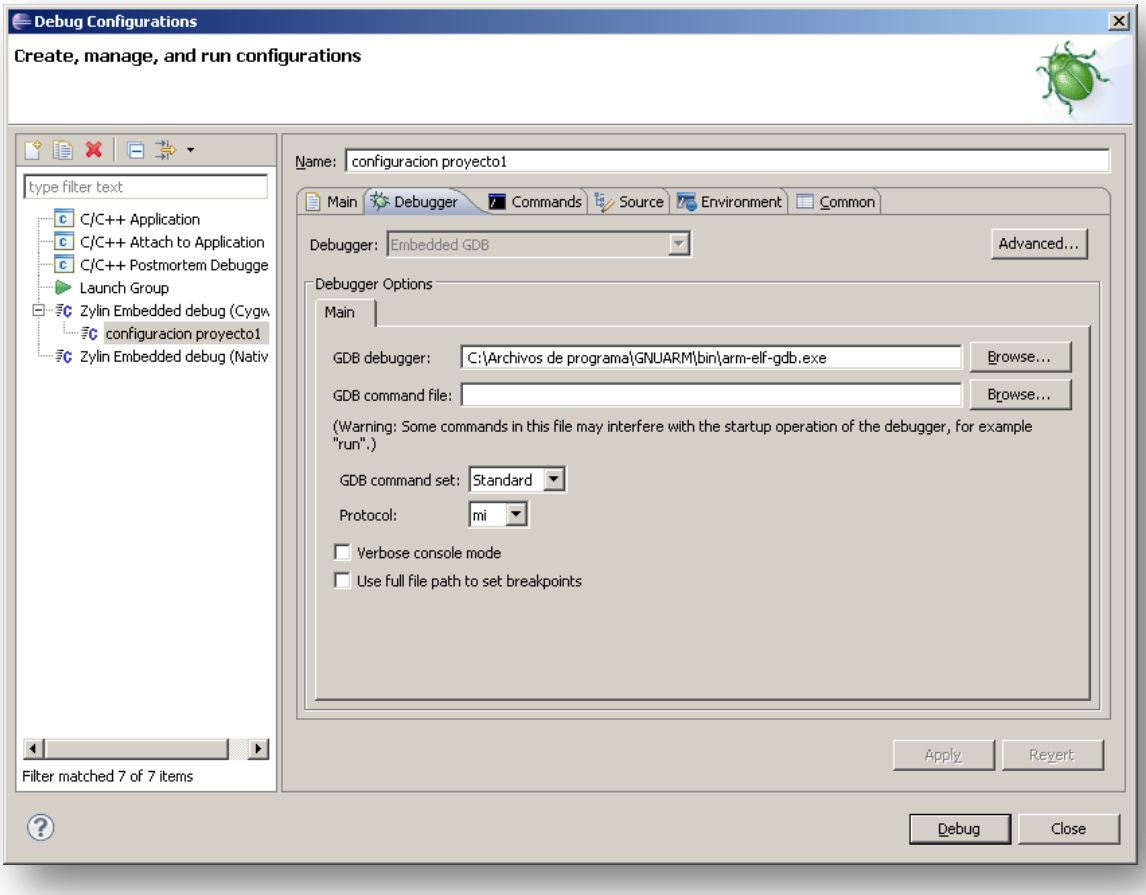
Una vez que se ha instalado el plugin hay que configurarlo, para ello, pulsamos sobre **"Zylin Embedded debug (Cygwin)"** pulsamos el símbolo  y aparecerá un nueva configuración del depurador.



En la ventana anterior hay que escribir en **“Name”:** El nombre que se le quiere asignar a la configuración. En **“Project”** el nombre del proyecto al que pertenece la configuración. En **“C/C++ Application”** mediante el botón **“Browse”** seleccionar la ruta en la que se encuentra el archivo main.out. De forma que tiene que quedar relleno de la siguiente forma:



Ahora si nos movemos a la pestaña **“debugger”,** en la etiqueta **“GDB Debugger”** hay que introducir la ruta en la que se encuentra la aplicación arm-elf-gdb. Esta aplicación se encuentra en la carpeta “bin” de la herramienta GNUARM. Además se debe borrar cualquier comando que se encuentre en la etiqueta **“GDB command file”** para que no interfiera en la comunicación con el J-Link GDB Server.



Por último, si pasamos a la siguiente pestaña (**"Commands"**), hay que introducir los comandos de inicialización y de ejecución. Para conocer los comandos, nos dirigimos al manual de ayuda del software del JTAG que se encuentra en "http://www.segger.com/cms/admin/uploads/productDocs/UM08005_JLinkGDBServer.pdf" [11]. Allí se abrirá el manual "JLinkGDBServer.pdf".

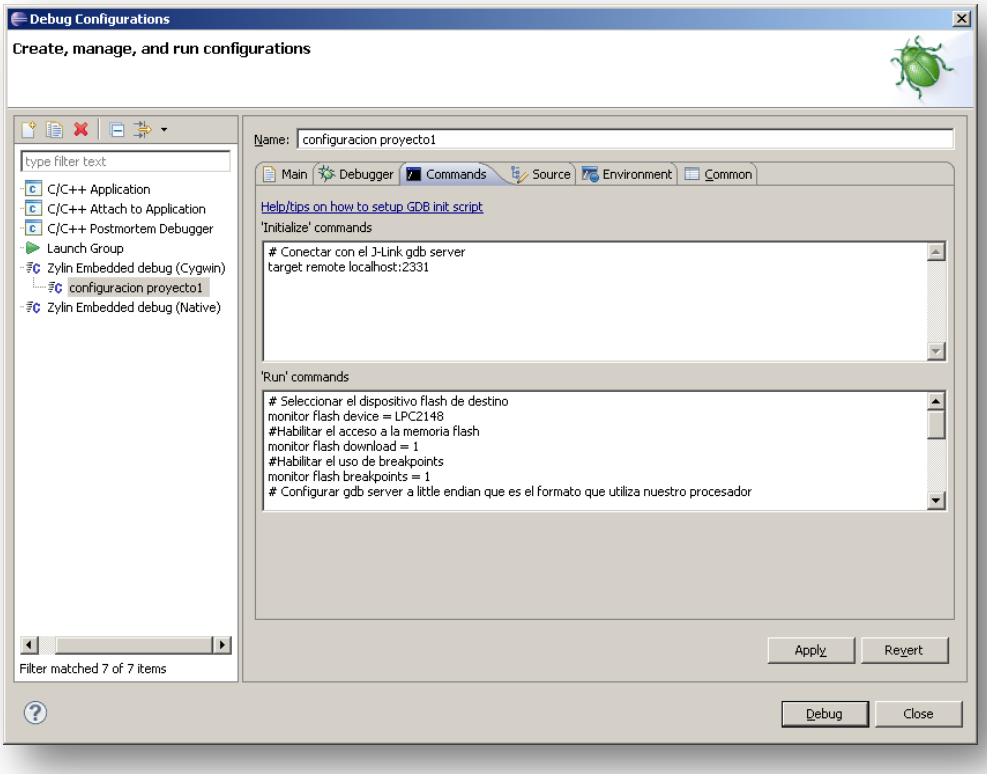
Para que Eclipse comunique correctamente con el procesador, los comandos de inicialización (**"Initialize commands"**) son:

```
# Conectar con el J-Link gdb server
target remote localhost:2331
```

Y los comandos de ejecución (**"run commands"**) son:

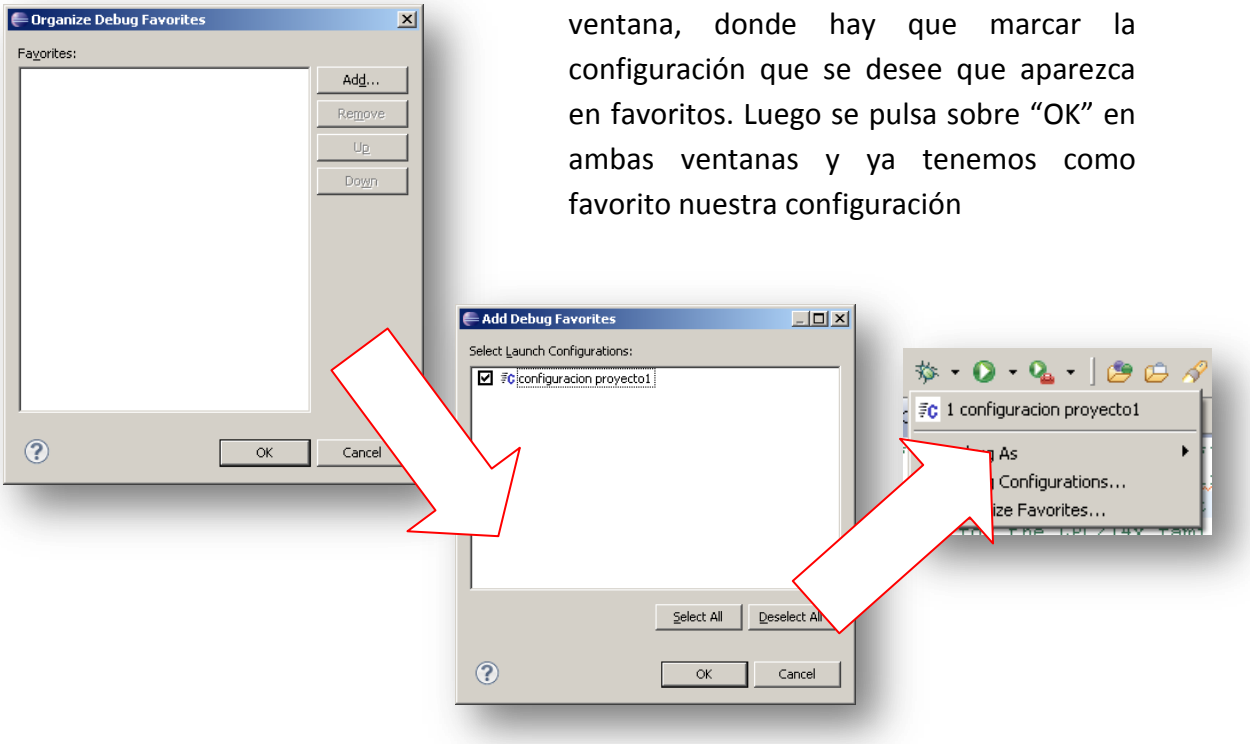
```
# Seleccionar el dispositivo flash de destino
monitor flash device = LPC2148
#Habilitar el acceso a la memoria flash
monitor flash download = 1
#Habilitar el uso de breakpoints
monitor flash breakpoints = 1
# Configurar gdb server a little endian que es el formato que utiliza nuestro procesador
monitor endian little
# Establecer la velocidad del JTAG como adaptativa
monitor speed adaptive
# Resetear el procesador
monitor reset 0
# Redireccionar a los primeros 64 bytes de la memoria flash, para ello se escribe 0x1 en el registro MMEMAP que
# tiene la dirección de memoria E01FC040.
monitor long 0xE01FC040 = 0x00000001
#establecer un breakpoint en el main
break main
#
load
continue
```

Finalmente la pestaña debug queda:



Al llegar a este punto, pulsamos sobre “Apply ” y después sobre “Close”.

Para que aparezca como favorito la configuración que se acaba de crear, hay que pulsar sobre  y después sobre **“Organize favorites”** se abrirá la siguiente ventana:

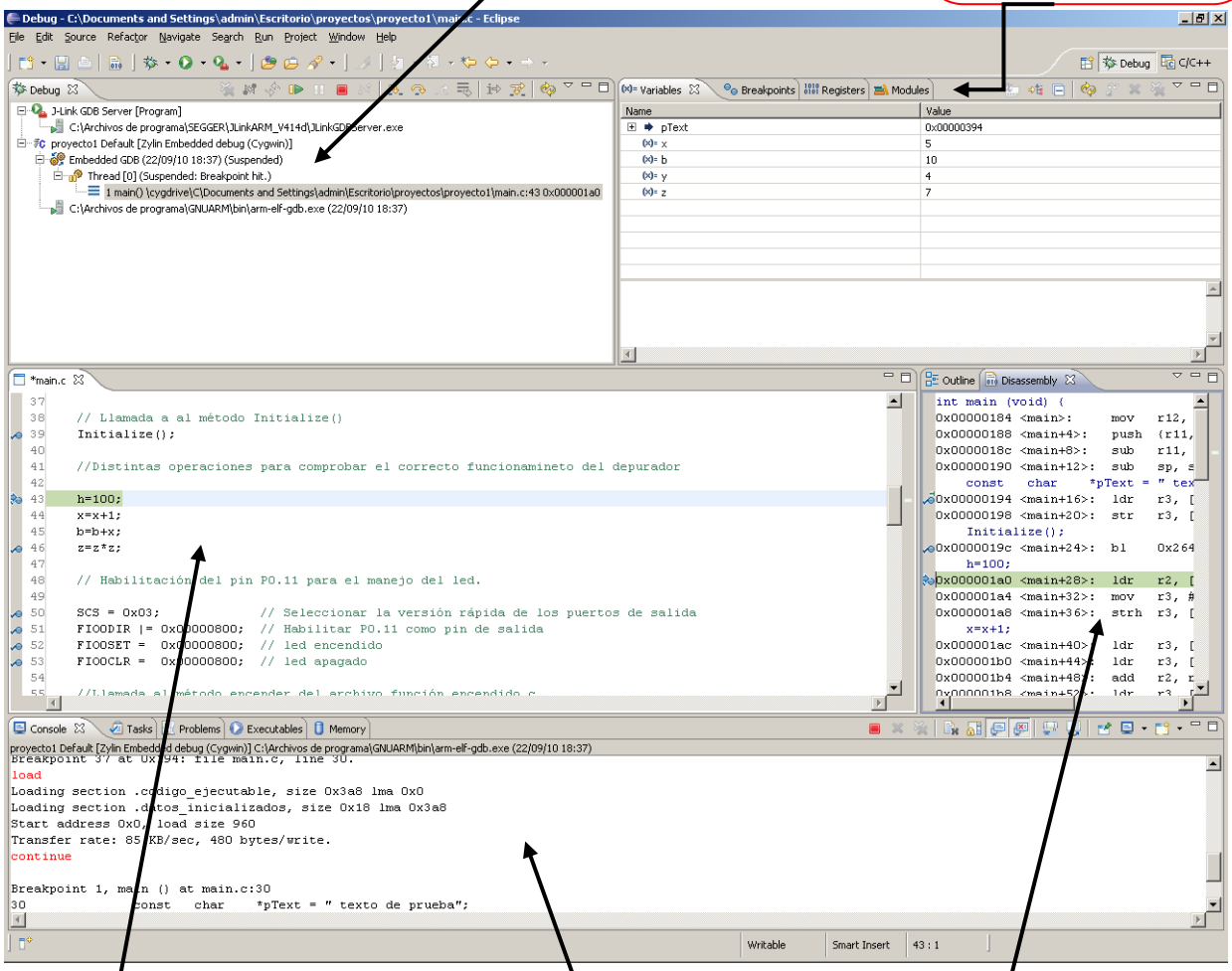


Al pulsar sobre “Add” se abre esta segunda ventana, donde hay que marcar la configuración que se desee que aparezca en favoritos. Luego se pulsa sobre “OK” en ambas ventanas y ya tenemos como favorito nuestra configuración

A partir de este punto es donde comienza verdaderamente el proceso de depurado de programas cargados en la memoria flash. Para que Eclipse abra la perspectiva de depuración, se pulsa sobre **“Window -> Open Perspective -> Debug”.**

Muestra información sobre la depuración y las herramientas que están siendo utilizadas, como por ejemplo J-Link GDB Server y arm-elf-gdb.exe

Las siguientes pestañas muestran: las variables, los breakpoints y el valor de los registros

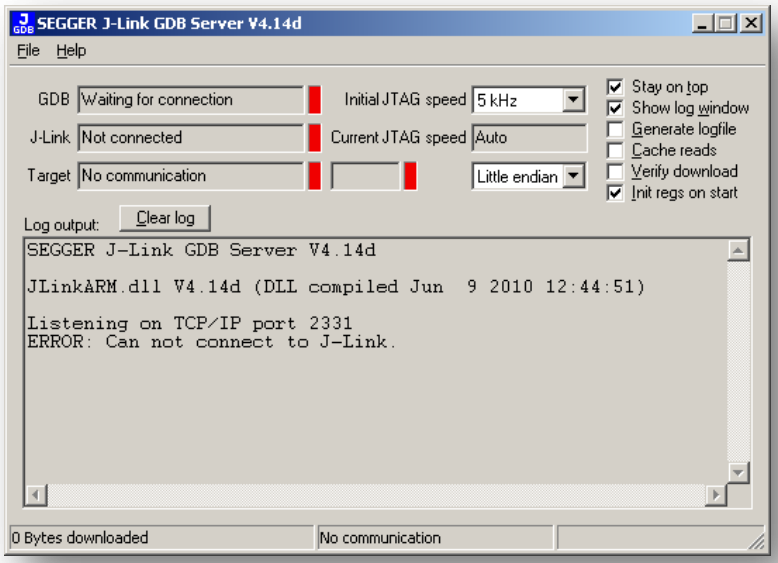


Muestra el código que está siendo ejecutado. La línea marcada es la siguiente línea a ejecutar. Los breakpoints están indicados en el margen izquierdo.

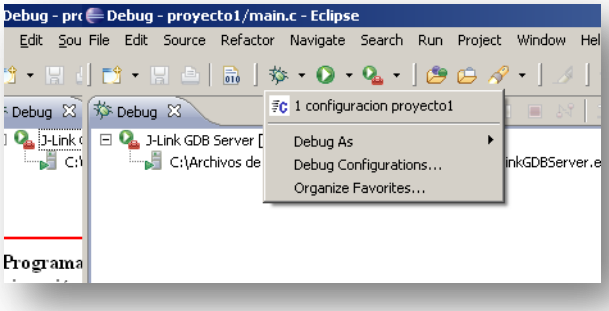
En este apartado se muestra el código que está siendo ejecutado pero en lenguaje ensamblador.

La consola muestra los comandos de configuración del JTAG y los errores que se puedan producir durante la depuración.

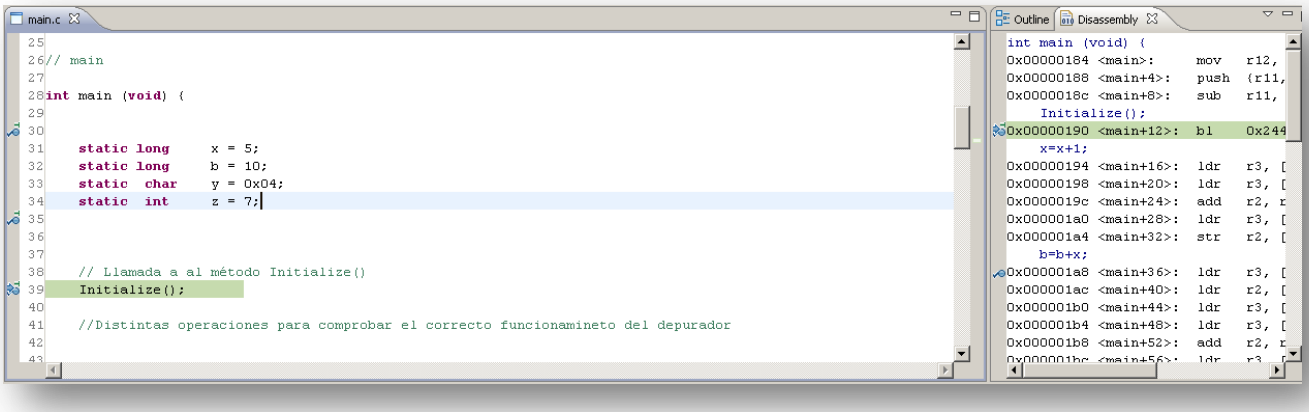
Para ejecutar el código, en la perspectiva debug, en primer lugar hay que abrir la herramienta J-Link GDB Server. Para ello, dirigirse a **"Run -> External Tools-> J-Link GDB Server"** y se abrirá la siguiente aplicación:



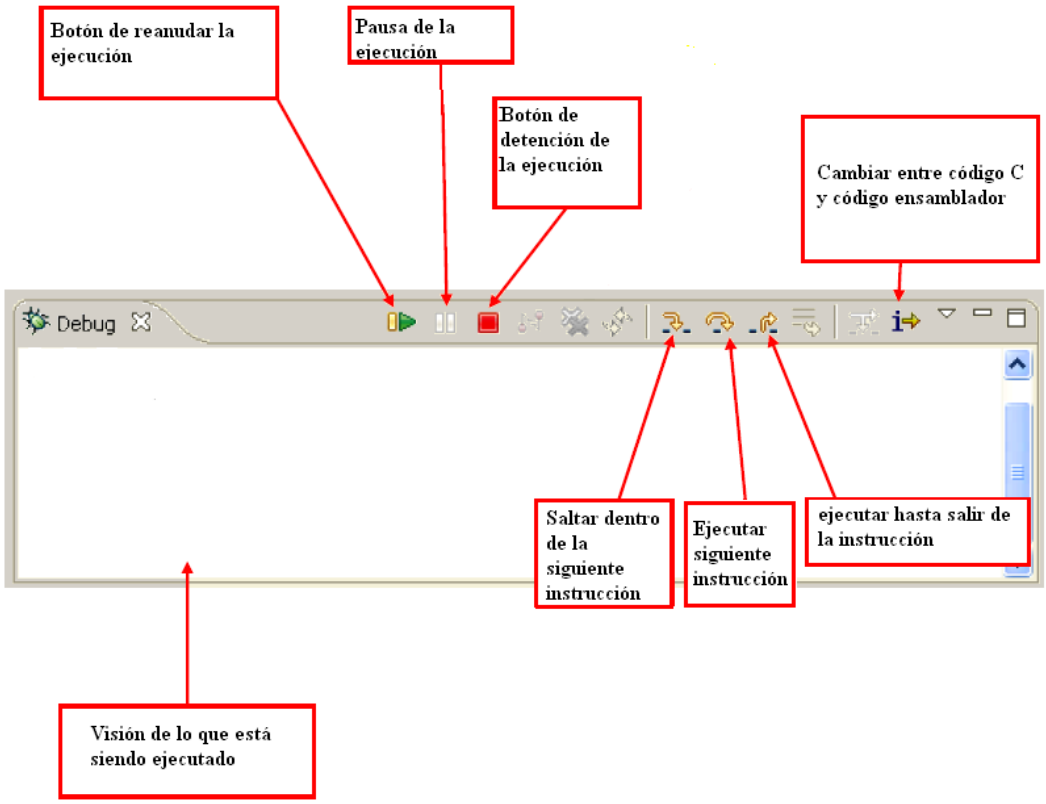
El siguiente paso consiste en lanzar el depurador que previamente se ha configurado, para ello, pulsar sobre  y después pulsar sobre la configuración que se había creado anteriormente:



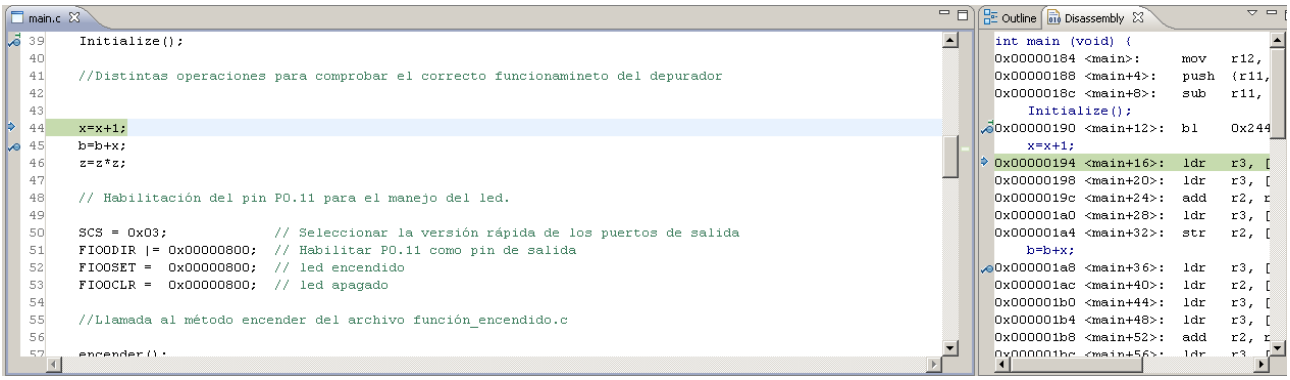
Una vez seleccionada la configuración, Eclipse establece la conexión con el procesador a través de la herramienta J-Link GDB Server y la ejecución del programa se queda parada en el main tal y como se había configurado en el depurador (la línea verde indica la próxima instrucción a ejecutar.)



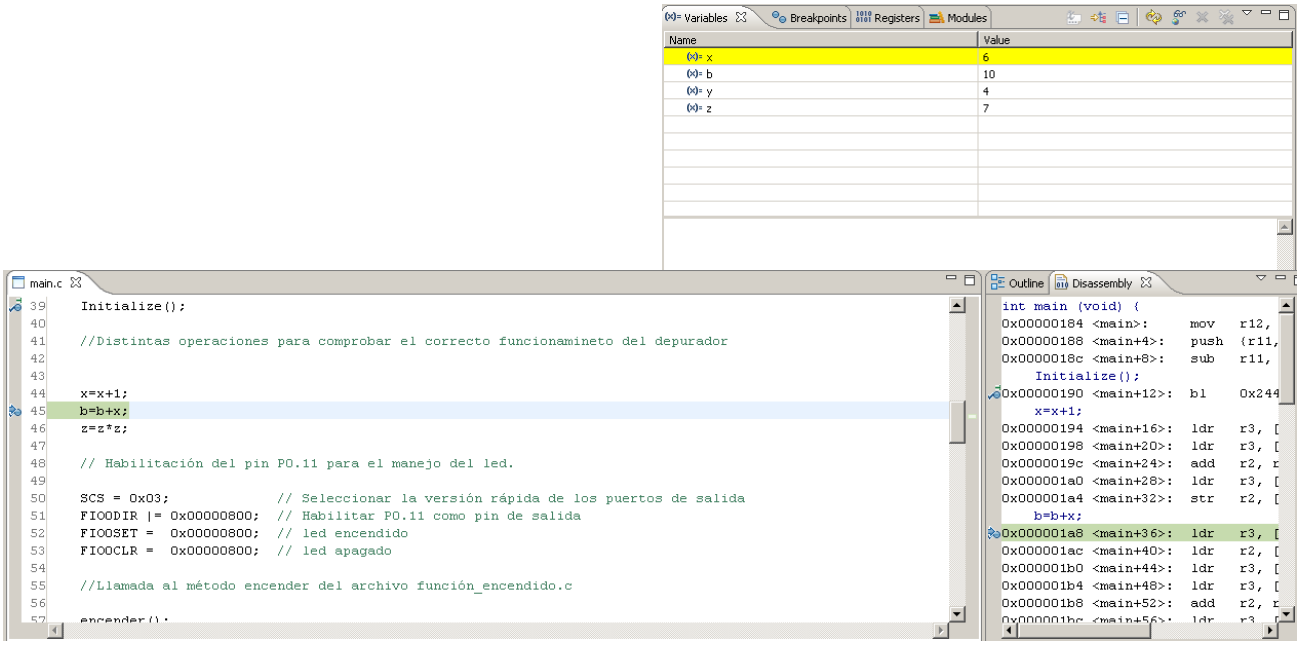
Para manejar la ejecución del programa en el depurador, hay que servirse de los siguientes botones:



Si se sigue con el proceso de depuración, al pulsar sobre  , se ejecuta el contenido de la función Initialize() pero el depurador no entra en ella y salta a la siguiente instrucción del main, que en este caso es x=x+1;.

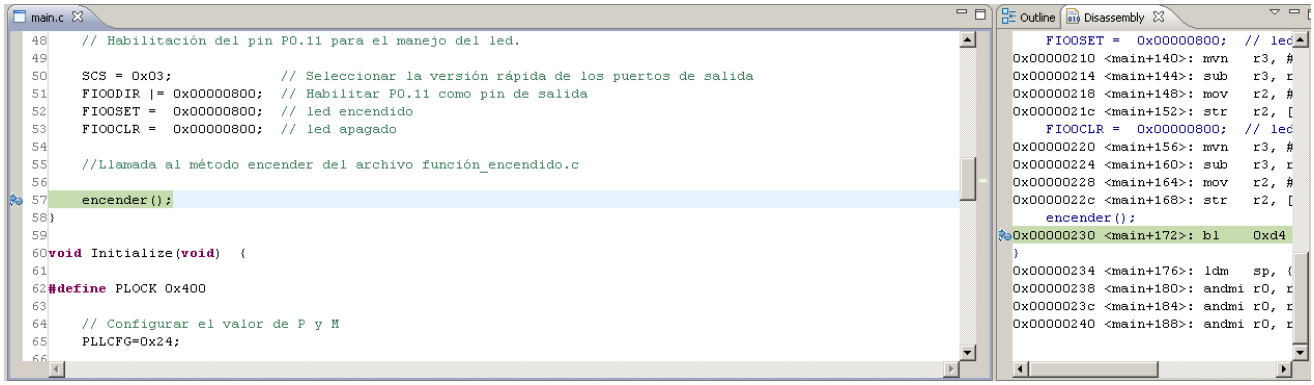


En el siguiente paso de depuración si se pulsa sobre  se ejecutará la instrucción x=x+1; y en la pestaña de variables se mostrará el nuevo valor adquirido por la variable x marcado de amarillo.

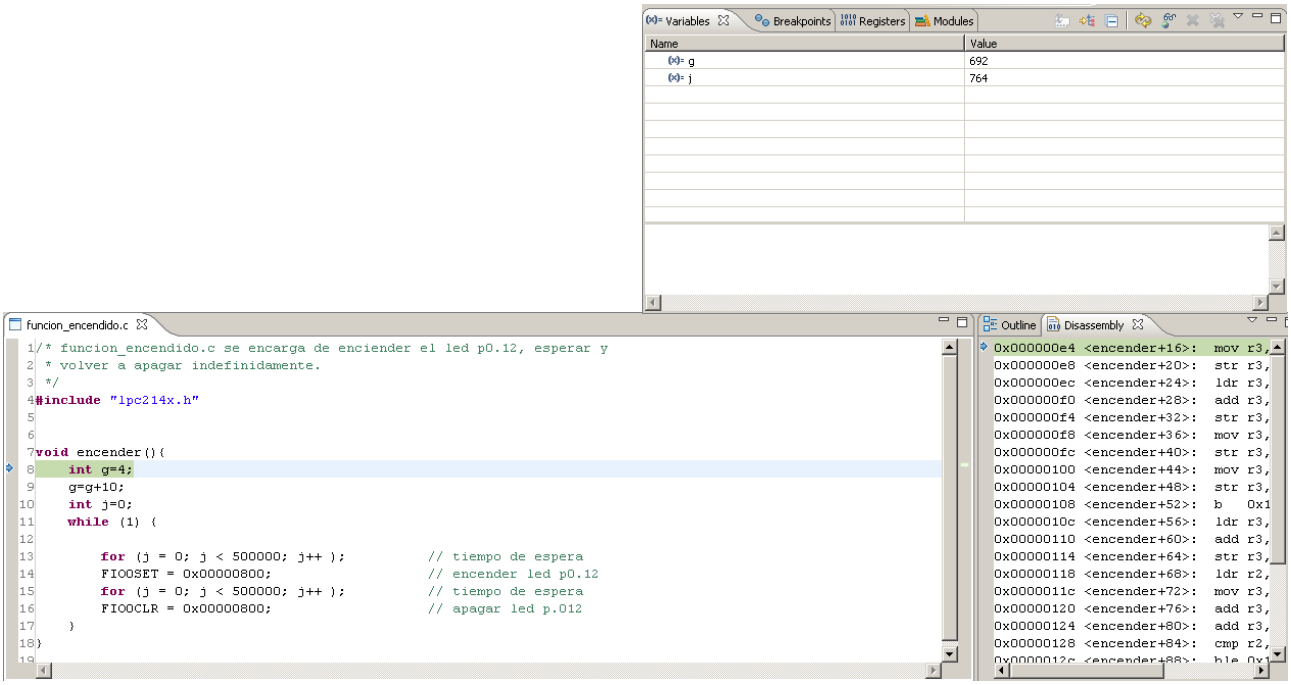


Ahora si se pulsa sobre  , la ejecución va a continuar hasta pararse en el siguiente breakpoint que en este caso se encuentra situado justo en la función

encender. Los breakpoints pueden ser añadidos durante la depuración, no es necesario ponerlos antes de comenzar la ejecución



En este momento si se pulsa sobre  , se accede al interior de la función encendido, donde se muestra su contenido y sus variables.



Ahora se puede seguir depurando paso a paso el programa y se puede comprobar como el led parpadea. Para finalizar la depuración pulsar sobre  y la depuración finalizará.

8. Conclusiones

El objetivo principal de este proyecto era el de sustituir el entorno de programación y depuración de Keil por otro que fuese de código libre y tras la finalización del proyecto se ha logrado.

El motivo principal por el que elegí este proyecto fue que siempre había oído hablar sobre código GNU pero nunca lo había tratado tan de cerca como lo he tratado en estos últimos meses y me parece que es una buena filosofía pues puedes llegar a crear tus propias herramientas.

El software profesional como por ejemplo el ofrecido por Keil es muy fácil de instalar y tienen soporte en caso de necesitarlo. Este software para el mundo profesional es el idóneo, pues aunque con herramientas GNU se ahorre en cuanto a las herramientas software, a la larga puede hacer perder dinero a la empresa debido a que los compiladores de keil son más fáciles de mantener y de manejar.

El hecho de que con la toolchain GNU no se pueda simular y con el software de Keil si, ya que ofrece dicha posibilidad, puede ser considerado una desventaja pero no es así, ya que al disponer de la placa y el JTAG que son de muy bajo coste comparados con otros que se pueden encontrar en el mercado, se puede ir depurando el programa en la propia placa sin necesidad de simular. De esta forma los posibles errores de simulación desaparecen, por lo que las posibles pruebas que se deseen realizar serán mucho más fiables.

Sin embargo, las herramientas GNU son una muy buena alternativa para un uso docente, ya que su coste es muy bajo y no son necesarias las prestaciones que ofrece el entono profesional. El peaje que hay que pagar por ello es el tener que adecuar los archivos del proyecto “Inicializar.s”, “linker_script.cmd”, “makefile” al tipo de procesador que se quiera utilizar, cuando en las herramientas comerciales simplemente hay que seleccionar el modelo del procesador y automáticamente el propio entorno de desarrollo se configura de tal forma que puede manejar el procesador seleccionado. Además de la instalación de las herramientas GNU es bastante más larga y complicada pero con la ayuda de este manual, espero que sea mucho más asequible y que se comprendan los conocimientos explicados.

9. Trabajo futuro

Durante la creación y puesta a punto de el toolchain GNU para el microprocesador LPC2148 nuevas ideas han surgido para mejorar y optimizar el toolchain, pero debido a los limitados recursos de tiempo disponibles no han podido llevarse a cabo en este proyecto. Las principales líneas de trabajo para continuar este proyecto son:

- Modificación de la toolchain para poder cargar los programas en la memoria RAM en lugar de la memoria flash.

- Utilizar el Toolset GNU Yagarto en lugar de Cygwin para así poder tener otra herramienta alternativa.

- Creación de una guía con unos pasos detallados sobre cómo modificar los ficheros “makefile”, “linker_scritp.cmd” y “Initialize.s” para que se pueda utilizar esta toolchain con otros procesadores que no sea el LPC2148.

- Creación de las rutinas de manejo de interrupciones del procesador, ya que en este proyecto, las interrupciones no están habilitadas.

- Crear un manual para poder programar el procesador mediante el puerto serie con la herramienta LPC2000 Flash Utiliy.

10. Diagrama de Gantt y presupuesto

10.1 Etapas del proyecto

Las etapas que se han realizado para la creación de este proyecto son :

Fase de planificación:

Análisis del objetivo: determinar cuál es el problema que hay que resolver, así como realizar una valoración global sobre el tema a desarrollar.

Estudio arquitectura procesador: conocer cómo se comporta el procesador y estudiar en profundidad las partes que sean necesarias para la realización del proyecto.

Evaluación de herramientas: una vez que se conoce cómo abordar el problema, se buscan las posibles herramientas que nos ayuden a resolverlo.

Elección de herramientas: tras conocer las distintas herramientas, hay que elegir según sus características, cual es la más apropiada para el proyecto.

Diseño del modelo operativo: una vez conocidas las herramientas que se van a utilizar, los objetivos marcados, y el código a desarrollar, se planifican las fases del desarrollo.

Fase de desarrollo:

Instalar herramientas: se inicia el proceso de instalación de las herramientas elegidas para el progreso del proyecto.

Desarrollo del código: se escribe el código necesario para que las herramientas seleccionadas interaccionen con el procesador.

Realizar pruebas con el procesador: se comprueba que el código cumple su cometido, en caso de que no, se vuelve a la fase anterior. Por ello estas dos etapas (desarrollo del código y la realización de pruebas con el procesador) se realizan en paralelo, corrigiendo cada fallo que se cometa.

Evaluación de resultados: una vez finalizado el código, se comprueba si cumple con los objetivos planteados al inicio del proyecto.

Análisis de prestaciones: aunque se cumplan los objetivos ¿es viable el proyecto?, es decir, no es un sistema lento, costoso, difícil de entender,…. En este punto es donde se evalúa si la elección de las herramientas ha sido errónea.

Optimización del código: consiste en preguntarse cómo poder mejorar el código para así mejorar el proyecto, cuestionarse si hay algo que no es necesario, , hacer un código menos extenso que así ocupe menos memoria en el procesador y haga que sea más rápida su ejecución.

Documentación de código: comentar el código.

Fase de redacción:

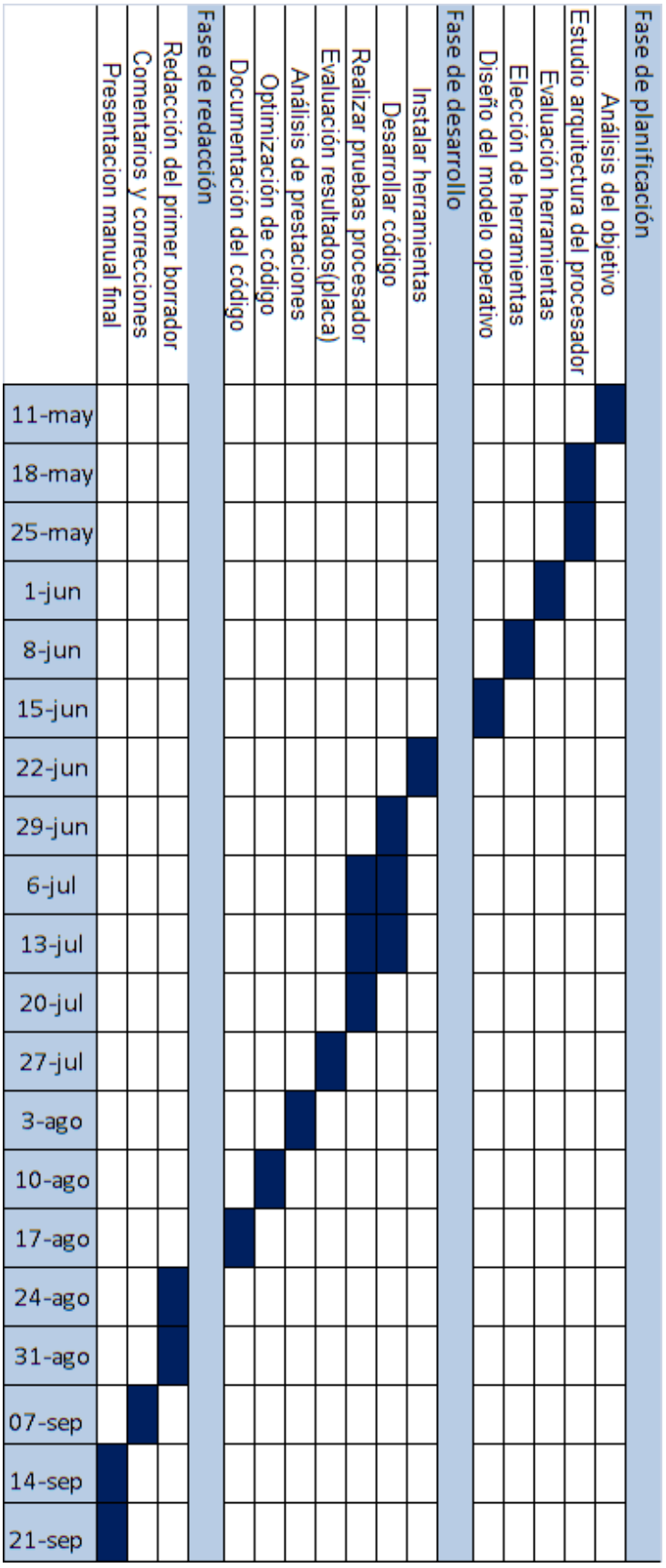
Redacción del primer borrador: se escribe el manual sobre el trabajo realizado y se explican las tareas ejecutadas.

Comentarios y correcciones: se repasa el primer borrador, y se estudia si es necesario cambiar algo o si hay algún fallo.

Presentación manual: se redacta el manual definitivo y se expone ante el cliente.

Cada fase así como sus distintas etapas, son realizadas por un equipo compuesto por dos ingenieros. Ambos llevan a cabo las distintas tareas y exponen sus preguntas y reflexiones sobre el proyecto, con el fin de finalizar el proyecto con éxito.

10.2 Diagrama de Gantt:



Ahora se presenta una tabla resumiendo la fecha de inicio y fin de de cada una de las tareas, su duración.

Actividad	Fecha de inicio	Duración	Fecha de fin
Fase de planificación			
Análisis del objetivo	11 de Mayo	1 semana	18 de Mayo
Estudio arquitectura del procesador	18 de Mayo	2 semanas	1 de Junio
Evaluación herramientas	1 de Junio	1 semana	8 de Junio
Elección de herramientas	8 de Junio	1 semana	15 de Junio
Diseño del modelo operativo	15 de Junio	1 semana	22 de Junio
Fase de desarrollo			
Instalar programas	22 de Junio	1 semana	29 de Junio
Desarrollar código	29 de Junio	4 semanas	27 de Julio
Realizar pruebas procesador	29 de Junio	4 semanas	27 de Julio
Evaluación resultados(placa)	27 de Julio	1 semana	03 de Agosto
Análisis de prestaciones	03 de Agosto	1 semana	10 de Agosto
Optimización del código	10 de Agosto	1 semana	17 de Agosto
Documentación del código	17 de Agosto	1 semana	24 de agosto
Fase de redacción			
Redacción del primer borrador	24 de Agosto	2 semanas	07 de Septiembre
Comentarios y correcciones	07 de Septiembre	1 semana	14 de Septiembre
Presentacion manual final	14 de Septiembre	2 semanas	21 de Septiembre

10.3 Presupuesto

El material necesario para la realización del proyecto ha sido el siguiente:

- 2 ordenadores.
- JTAG.
- Placa test LPC2148.
- Java JRE.
- Eclipse.
- Cygwin.
- GNUARM.
- Software de SEGGER.
- Zylin cdt plugin.
- Material de oficina.

Cómo este proyecto consistía en utilizar herramientas GNU, el coste del software utilizado es cero. Por lo que el único coste material que ha tenido el proyecto es el de los ordenadores, JTAG y la placa del LPC2148 cuyos costes se detalla a continuación.

EQUIPOS					
Descripción	Coste (Euro)	% Uso dedicado proyecto	Dedicación (meses)	Periodo de depreciación	Coste imputable [d]
2 Ordenadores	1.600,00	100	5	20	400,00
1 JTAG	85,00	100	2	48	3,54
1 PLACA LPC2148	40,00	100	2	30	2,67
				Total	406,21

[d] Fórmula de cálculo de la Amortización:

$\frac{A}{B} x C x D$

A = nº de meses desde la fecha de facturación en que el equipo es utilizado
B = periodo de depreciación
C = coste del equipo (sin IVA)
D = % del uso que se dedica al proyecto

En cuanto al coste del personal, suponiendo un salario de 1.800€ al mes el sueldo de un Ingeniero, durante un periodo de 5 meses:

PERSONAL					
Apellidos y nombre	N.I.F. (no rellenar - solo a titulo informativo)	Categoría	Dedicación (hombres mes) [a]	Coste hombre mes	Coste (Euro)
Rivera Campos, Alonso	50411125R	Ingeniero	5	1.800,00	9.000,00
		Hombres mes 5		Total	9.000,00

[a] 1 Hombre mes = 150,00 horas. Máximo anual de dedicación de 2 hombres mes (300 horas)
Máximo anual para PDI de la Universidad Carlos III de Madrid de 8,8 hombres mes (1.320 horas)

A los costes anteriores hay que añadir otros costes adicionales al proyecto, como es el material de oficina empleado para la creación del manual.

OTROS COSTES DIRECTOS DEL PROYECTO		
Descripción	Empresa	Costes imputable
Material de oficina	Folder	100,00
	Total	100,00

Por lo que el presupuesto total de este proyecto asciende a la cantidad de **nueve mil quinientos seis** euros.

Referencias

[1] Página de inicio de Java: http://java.sun.com. Accedido en Junio 2010.

[2] Página de Eclipse: http://www.eclipse.org/. Accedido en Junio 2010.

[3] Página de CYGWIN: http://www.cygwin.com. Accedido en Junio 2010.

[4] Página de GNUARM: http://www.gnuarm.com. Accedido en Junio 2010.

[5] Using the GNU Compiler Collection. Disponible en Internet: http://gcc.gnu.org/onlinedocs/gcc-4.5.0/gcc.pdf. Accedido en Septiembre 2010.

[6] Manual on-line del Linker GNU. Disponible en Internet: http://sourceware.org/binutils/docs-2.20/ld/. Accedido en Septiembre 2010.

[7] Manual on-line de linker_scripts. Disponible en Internet: http://sourceware.org/binutils/docs-2.20/ld/Scripts.html#Scripts. Accedido en Septiembre 2010

[8] Manual on-line de la herramienta objcopy. Disponible en Internet: http://sourceware.org/binutils/docs/binutils/objcopy.html. Accedido en Septiembre 2010.

[9] Make Un programa para controlar la recompilación. Disponible en Internet: http://www.uca.es/softwarelibre/publicaciones/make.pdf. Accedido en Septiembre 2010.

[10] Manual J-Flash ARM. Disponible en Internet: http://www.segger.com/cms/admin/uploads/productDocs/UM08003_JFlashARM.pdf. Accedido en Septiembre 2010

[11] Manual J-Link GDB Server. Disponible en Internet: http://www.segger.com/cms/admin/uploads/productDocs/UM08005_JLinkGDBServer.pdf. Accedido en Septiembre 2010

[12] Página de inicio de SEGGER: http://www.segger.com. Accedido en Junio 2010.

[13] Cygwin User Guide. Disponible en Internet: http://www.cygwin.com/cygwin-ug-net/cygwin-ug-net.html. Accedido en Septiembre 2010.

[14] Registros en modo thumb y ARM. Disponible en Internet: http://ftp.frh.utn.edu.ar/electronica/materias/td2/publico/ARM/ARM_Registros_en_modo_THUMB_y_ARM.doc. Accedido en Septiembre 2010.

[15] Set de instrucciones de ensamblador. Disponible en Internet: http://www.heyrick.co.uk/assembler/qfinder.html. Accedido Septiembre 2010.

[16] *Manual de usuario de la familia de procesadores LPC214x*. Disponible en Internet: http://www.nxp.com/documents/user_manual/UM10139.pdf.Accedido en Septiembre 2010.

[17] ARM 7TDMI Data Sheet. Disponible en Internet: http://www.eecs.umich.edu/~tnm/power/ARM7TDMIvE.pdf. Accedido en Septiembre 2010.

[18] Trevor Martin. "*The Insiders Guide To The Philips ARM7- Based Microcontrollers*". Hitex. Segunda revisión Septiembre 2006.

[19] Using as: The GNU Assembler V2.14. Disponible en Internet: http://www.gnuarm.com/pdf/as.pdf. Accedido en Septiembre 2010. Accedido en Septiembre 2010.

[20] Manual on-line de la herramienta objdump. Disponible en Internet: http://sourceware.org/binutils/docs/binutils/objdump.html. Accedido en Septiembre 2010.

[21] Página de inicio Zylin. Dsiponible en Internet: http://www.zylin.com/. Accedido en Septiembre 2010.

[22] Tutorial ARM Eclipse. Disponible en Internet: http://www.olimex.cl/tutorial/Tutorial%20ARM%20rev2.pdf. Accedido en Junio 2010.